

```

0100      0001  EXP - A monitor for the Expander
0100      0003  ;
0100      0004  (C) 1981 by Micro-Expander, Inc.
0100      0005  All Rights Reserved
0100      0006  ;
0100      0007
0100      0008 ROM EQU 0F000H ; the starting address of the monitor rom
0100      0009 P1BIAS EQU 00800H ; bias to second page of rom
0100      0010 P1ORG EQU ROM+00800H ; where to start page 1
0100      0011 ENDP1 EQU P1ORG+00800H ; end of page 1
0100      0012
0100      0013 ORG ROM
0100      0014
F000      0015 JP RESET ; Monitor Cold Start, reset everything
F003      0016 JP RESTART ; Monitor Warm Start, reset the stack only
F006      0017
F006      0018 JP KSTAT ; returns Offh if char available at kbd else 0
F009      0019 JP KBREAD ; returns a character from the keyboard in acc
F00C      0020 JP PUTCHAR ; Print the character in c on the video
F00F      0021
F00F      0022 JP PSTAT ; same as kstat except with parallel port
F012      0023 JP PREAD ; same as kread " " " "
F015      0024 JP POUT ; same as putchar " " " "
F018      0025
F018      0026 JP SERSTAT ; serial status.
F01B      0027 JP SERIN ; serial input
F01E      0028 JP SEROUT ; same as putchar
F021      0029
F021      0030 JP SERSEL ; select serial baud rate
F024      0031
F024      0032 JP CASSTAT ; cassette status
F027      0033 JP CASIN ; cassette input
F02A      0034 JP CASOUT ; cassette output
F02D      0035
F02D      0036 JP RFILE ; read a cassette file
F030      0037 JP WFILE ; write a cassette file
F033      0038 JP RBLOCK ; read a block of data
F036      0039 JP WBLOCK ; write a block of data
F039      0040 JP SYNC ; cassette sync on header
F03C      0041 JP WSYNC ; cassette write header
F03F      0042 JP RTAIL ; end cassette read
F042      0043 JP WTAIL ; end cassette write
F045      0044 ;
F045      0045 START LD SP,MSTACK
F048      0046 LD C,CLEAR ; clear the screen
F04A      0047 CALL MPUTC
F04D      0048
F04D      0049 RESTART LD SP,MSTACK
F050      0050 EI ; turn interrupts on
F051      0051
F051      0052 CLOOP LD C,' ' ; out prompt
F053      0053 CALL MPUTC ; c3 0040
F056      0054
F056      0055 CALL GETLINE
F059      0056
F059      0057 LD HL,(CURPOS)
F05C      0058 LD A,(WASTHERE)
F05F      0059 LD (HL),A
F060      0060 CALL ATOLC

```

F063 0F 01	0061	LD	C,1	
F065 CD 38 F5	0062	CALL	LCTOA	
F068 0A 06	0063	LD	B,6	
F06A CD 20 F3	0064	CALL	FINDNB	
F06D 7F	0065	LD	A,(HL)	
F06E F6 20	0066	OR	020H	
F070	0067			
F070 16 0B	0068	LD	D,ECMDTAB-CMDTAB/3	
F072 CD 76 F5	0069	CALL	SWITCH	
F075	0070			
F075 64	0071	CMDTAB DB	'd'	
F076 AF F0	0072	DW	DUMP	
F078 65	0073	DB	'e'	
F079 A4 F0	0074	DW	ENTER	
F07B 6A	0075	DB	'J'	
F07C 08 F1	0076	DW	JUMP	
F07E 6E	0077	DB	'n'	
F07F 00 E8	0078	DW	0E800H	; North Star boot address, allows 59K CP/M
F081 6D	0079	DB	'm'	; both controllers can co-exist this way.
F082 00 EC	0080	DW	0EC00H	; Micropolis boot address, allows 59K CP/M.
F084 74	0081	DB	't'	
F085 1E F1	0082	DW	TERM	
F087 73	0083	DB	's'	
F088 3B F1	0084	DW	CSAVE	
F08A 6C	0085	DB	'l'	
F08B 41 F1	0086	DW	CLOAD	
F08D 62	0087	DB	'b'	
F08E DF F6	0088	DW	SELBAUD	
F090 63	0089	DB	'c'	
F091 4E F1	0090	DW	CAT	
F093 70	0091	DB	'P'	
F094 14 F1	0092	DW	PDLOAD	
F096	0093	ECMDTAB EQU	\$	
F096	0094			
F096 36 3F	0095	ERROR LD	(HL),??	
F098	0096			
F098 CD D7 F2	0097	NXTCMD CALL	CRLF	
F09B 18 B4	0098	JR	CLOOP	
F09D	0099			
F09D	0100			
F09D	0101		Software reset point	
F09D	0102			
F09D	0103			
F09D F3	0104	RESET DI		
F09E FD 21 06 F0	0105	LD	IY,INIT-P1BIAS	
F0A2 18 04	0106	JR	EDJCAL	;this will call page 1, but not return
F0A4	0107			
F0A4	0108	COPY	EDJ.CODE/1	
F0A4	0109			
F0A4	0110	ENTER	- the enter command (page 0 to page 1 linkage)	
F0A4	0111			
F0A4	0112		command syntax: e <addr>	
F0A4	0113			
F0A4	0114			
F0A4 FD 21 9B F4	0115	ENTER LD	IY,P1ENTER-P1BIAS	
F0A8	0116			
F0A8 CD D3 F7	0117	EDJCAL CALL	POTP1	
F0AE 38 E9	0118	JR	C.ERROR	
F0AD 18 E9	0119	JR	NXTCMD	
F0AF	0120			

```

FOAF 0121 ;
FOAF 0122 ; DUMP - the dump command
FOAF 0123 ;
FOAF 0124 ; command syntax: d [start_addr] [end_addr]
FOAF 0125 ;
FOAF 0126
FOAF CD ED F2 0127 DUMP CALL GETADDR ; get start address to de
FOB2 3F E2 0128 JR C,ERROR
FOB4 05 0129 PUSH DE ; save it
FOB5 CD ED F2 0130 CALL GETADDR ; get ending address to...
FOB9 EB 0131 EX DE,HL ; ...hl
FOB9 D1 0132 POP DE ; retrieve start address
FOBA 0133
FOBA CD 07 F2 0134 DU1 CALL CRLF ; print address
FOBD 7A 0135 LD A,D ; print hi byte
FOBE CD 48 F3 0136 CALL PUTHB
FOC1 7B 0137 LD A,E ; print lo byte
FOC2 CD 48 F3 0138 CALL PUTHB
FOC5 0139
FOC5 D5 0140 PUSH DE ; save it
FOC6 06 10 0141 LD B,16 ; print 16 bytes
FOC8 0E 20 0142 DU2 LD C, ; print a space
FOCA CD DE F2 0143 CALL MPUTC
FOCD 1A 0144 LD A,(DE) ; fetch the byte
FOCE CD 48 F3 0145 CALL PUTHB ; print it
FOD1 13 0146 INC DE ; increment pointer
FOD2 10 F4 0147 DJNZ DU2 ; and repeat
FOD4 0148
FOD4 0E 20 0149 LD C, ; space over a little
FOD6 CD DE F2 0150 CALL MPUTC
FOD9 CD DE F2 0151 CALL MPUTC
FODC 0152
FODC D1 0153 POP DE ; retrieve the address
FODD 06 10 0154 LD B,16 ; print this many bytes
FODF 1A 0155 DU3 LD A,(DE) ; fetch the byte
FOE0 FE 20 0156 CP ; check for printable ascii
FOE2 38 04 0157 JR C,DU4 ; jump if not
FOE4 FE 7F 0158 CP 07FH ; check for legal ascii
FOE6 38 02 0159 JR C,DU5 ; jump if so
FOE8 3E 2E 0160 DU4 LD A, '.' ; everything else gets a period
FOEA 4F 0161 DU5 LD C,A ; character to c
FOEB CD DE F2 0162 CALL MPUTC
FOEE 13 0163 INC DE ; increment the pointer
FOEF 10 EE 0164 DJNZ DU3 ; repeat
FOF1 0165
FOF1 7D 0166 LD A,L ; check for end of dump
FOF2 93 0167 SUB A,E
FOF3 7C 0168 LD A,H
FOF4 9A 0169 SBC A,D
FOF5 38 0F 0170 JR C,EXTEDJ ; jump if so
FOF7 0171
FOF7 CD 53 F7 0172 CALL KSTAT ; else check for stop
FOFA 28 BE 0173 JR Z,DU1
FOFC CD 58 F7 0174 CALL KBREAD ; burn the first character
FOFF CD 58 F7 0175 CALL KBREAD ; wait for the next
F102 FE 1B 0176 CP 01BH ; the escape key aborts us
F104 20 B4 0177 JR NZ,DU1
F106 18 90 0178 EXTEDJ JR NXTCMD
F108 0179 ;
F108 0180 ;

```



```

F108      0181 ; JUMP - Jump to an address, a RET will get you back
F108      0182 ;
F108      0183 ; command syntax: J Caddr2
F108      0184 ;
F108      0185
F108 CD ED F2      0186 JUMP CALL GETADDR      ; get the address
F108 38 89          0187 JR C,ERROR      ; Jump if error
F10D EB            0188 EX DE,HL      ; address to hl
F10E CD 13 F1      0189 CALL HLJUMP      ; do the call
F111 18 F3          0190 JR EXTEDJ      ; get the next command
F112 E9            0191 HLJUMP JP (HL)
F114            0192 ;
F114            0193
F114            0194 COPY PDL.CODE/1
F114            0195 ;
F114            0196 ; Down Load to memory from external device via parallel inport
F114            0197 ;
F114            0198 ; command syntax: p
F114            0199 ;
F114            0200 ; The data transfer format provided by the transmitting device
F114            0201 ; should be as follows:
F114            0202 ;
F114            0203 ; 8 bytes - 00h (null)
F114            0204 ; 1 " A5h sync character
F114            0205 ; 2 " load address
F114            0206 ; 2 " byte count
F114            0207 ; 1 " checksum for header
F114            0208 ; n " data to be transmitted (8 bits w/o parity)
F114            0209 ; 1 " checksum for data
F114            0210 ;
F114            0211
F114 FD 21 39 F7    0212 PDLOAD LD IV,DLOAD-P1BIAS
F118 CD D3 F7      0213 CALL POTP1
F118 C3 98 F0      0214 JP NXTCMD
F11E            0215 ;
F11E            0216 COPY TRM.CODE/1
F11E            0217 ;
F11E            0218 ; TERM - Something of a terminal command
F11E            0219 ; Nothing fancy, a NUL ( control-@ ) will get you out
F11E            0220 ;
F11E            0221 ; command syntax: t
F11E            0222 ;
F11E            0223
F11E CD 53 F7      0224 TERM CALL KSTAT      ; anything from the keyboard?
F121 28 0A          0225 JR Z,TERM1      ; try the serial port if not
F123            0226
F123 CD 58 F7      0227 CALL KBREAD      ; else get the character
F126 B7            0228 OR A      ; check for NUL
F127 28 00          0229 JR Z,EXTEDJ      ; quit if so
F129            0230
F129 4F            0231 LD C,A      ; else send to the serial port
F12A CD 20 F7      0232 CALL SEROUT
F12D            0233
F12D CD 01 F7      0234 TERM1 CALL SERSTAT      ; anything at serial port?
F130 28 EC          0235 JR Z,TERM      ; Jump if not
F132            0236
F132 CD 0A F7      0237 CALL SERIN      ; else get it...
F135 4F            0238 LD C,A
F136 CD 0E F2      0239 CALL MOUTC      ; ...to the screen
F138 1C E3          0240 JR TERM

```


F13B	0241 ;
F13B	0242 COPY CAS.CODE/1
F13B	0243 ;
F13B	0244 ; CSAVE - Write a block of data to the cassette
F13B	0245 ;
F13B	0246 ; command syntax: s <name> <start_addr> <end_addr>
F13B	0247 ;
F13B	0248
F13B FD 21 F0 F4	0249 CSAVE LD IY,P1CSAVE-P1BIAS
F13F 18 04	0250 JR CCALO
F141	0251 ;
F141	0252 ;
F141	0253 ; CLOAD - Load a block from the cassette
F141	0254 ;
F141	0255 ; command syntax: L [<name> [<load_addr>]]
F141	0256 ;
F141	0257
F141 FD 21 AE F6	0258 CLOAD LD IY,P1CLOAD-P1BIAS
F145	0259 ;
F145 CD D3 F7	0260 CCALO CALL POTP1
F148 DA 96 F0	0261 JP C.ERROR
F14B C3 98 F0	0262 JP NXTCMD
F14E	0263
F14E	0264 ;
F14E	0265 ; CAT - Catalogue a tape
F14E	0266 ;
F14E	0267
F14E FD 21 5E F5	0268 CAT LD IY,P1CAT-P1BIAS
F152 CD D3 F7	0269 CALL POTP1
F155 C3 98 F0	0270 JP NXTCMD
F158	0271
F158	0272 ;
F158	0273 ; RFILE - read a file
F158	0274 ;
F158	0275
F158 FD 21 CE F5	0276 RFILE LD IY,P1RFILE-P1BIAS
F15C	0277 ;
F15C CD D3 F7	0278 CCAL1 CALL POTP1
F15F C9	0279 RET
F160	0280
F160	0281 ;
F160	0282 ; WFILE - Write a file
F160	0283 ;
F160	0284
F160 FD 21 99 F5	0285 WFILE LD IY,P1WFILE-P1BIAS
F164 18 F6	0286 JR CCAL1
F166	0287
F166	0288 ;
F166	0289 ; WBLOCK - Write a block of data
F166	0290 ;
F166	0291 ; entry: hl has address of the block
F166	0292 ; de has count
F166	0293 ;
F166	0294 ; exit: if carry is reset the block was written
F166	0295 ; else carry is set and the abort key was seen
F166	0296 ;
F166	0297
F166 ED 73 88 FE	0298 WBLOCK LD (MSP),SP
F16A 31 80 FF	0299 LD SP,ASTACK
F16D 06 00	0300 LD B,0 ; init the checksum

F16F		0301			
F16F CD 40 F2		0302 WB1	CALL	CHKABORT	
F172 38 0D		0303	JR	C, WB2	
F174		0304			
F174 4E		0305	LD	C, (HL)	; get next byte
F175 79		0306	LD	A, C	; compute the checksum
F176 80		0307	ADD	A, B	
F177 47		0308	LD	B, A	; save the new checksum
F178		0309			
F178 CD A0 F7		0310	CALL	CASOUT	; write the data byte
F17B 23		0311	INC	HL	; point to the next byte...
F17C 1B		0312	DEC	DE	; decrement count
F17D 7B		0313	LD	A, E	; check for zero
F17E B2		0314	OR	D	
F17F 20 EE		0315	JR	NZ, WB1	; JUMP if count != 0
F181		0316			
F181 F5		0317 WB2	PUSH	AF	; save carry
F182 48		0318	LD	C, B	; else write the checksum
F183 CD A0 F7		0319	CALL	CASOUT	
F186 F1		0320	POP	AF	; restore carry
F187 ED 7B B8 FE		0321	LD	SP, (MSP)	
F18B C9		0322	RET		
F18C		0323 ;			
F18C		0324 ;			
F18C		0325 ;	RBLOCK	- Read a block of data from the cassette	
F18C		0326 ;			
F18C		0327 ;	entry:	de has byte count	
F18C		0328 ;		hl has load address	
F18C		0329 ;			
F18C		0330 ;	exit:	carry is set if abort key was hit	
F18C		0331 ;		zero flag is set if checksum is ok	
F18C		0332 ;			
F18C		0333			
F18C ED 73 B8 FE		0334 RBLOCK	LD	(MSP), SP	
F190 31 80 FF		0335	LD	SP, ASTACK	
F193 06 00		0336	LD	B, 0	; init check sum
F195		0337			
F195 CD 40 F2		0338 RB1	CALL	CHKABRT	; check for abort key
F198 38 22		0339	JR	C, RB2	
F19A CD 81 F7		0340	CALL	CASSTAT	
F19D B7		0341	OR	A	
F19E 28 F5		0342	JR	Z, RB1	
F1A0 CD 8A F7		0343	CALL	CASIN	; get the next byte of data
F1A3 77		0344	LD	(HL), A	; PLOP it into memory
F1A4 80		0345	ADD	A, B	; compute checksum
F1A5 47		0346	LD	B, A	; save it
F1A6 23		0347	INC	HL	; increment pointer
F1A7		0348			
F1A7 1B		0349	DEC	DE	; decrement count
F1A8 7B		0350	LD	A, E	
F1A9 B2		0351	OR	D	
F1AA 20 E9		0352	JR	NZ, RB1	; JUMP if not enough yet
F1AC		0353			
F1AC CD 40 F2		0354 RB3	CALL	CHKABRT	
F1AF 38 0B		0355	JR	C, RB2	
F1B1 CD 81 F7		0356	CALL	CASSTAT	
F1B4 B7		0357	OR	A	
F1B5 28 F5		0358	JR	Z, RB3	
F1B7 CD 8A F7		0359	CALL	CASIN	; else get the check sum
F1BA 90		0360	SUB	A, B	; check if ok

F1B8 B7	0361	OR	A	
F1BC 18 7D	0362	RB2	JR	SYNC4
F1BE	0363			
F1BE	0364			
F1BE	0365			WSYNC - Write the sync header for the cassette
F1BE	0366			
F1BE	0367			It consists of 128 NULL's followed by an 0A5H
F1BE	0368			sync byte
F1BE	0369			
F1BE	0370			
F1BE ED 73 B8 FE	0371	WSYNC	LD	(MSP),SP
F1C2 31 80 FF	0372		LD	SP,ASTACK
F1C5 F3	0373		DI	
F1C6 E5	0374		PUSH	HL
F1C7 CD 63 F2	0375		CALL	CINIT ; do global cassette initialization
F1CA 21 99 F3	0376		LD	HL,CXTIMER-P1BIAS ; set it to cassette xmit timer
F1CD 22 F2 FF	0377		LD	(INTV1),HL
F1D0 21 69 F0	0378		LD	HL,SPEINT-P1BIAS ; set edge interrupt to dummy handler
F1D3 22 F4 FF	0379		LD	(INTV2),HL
F1D6 E1	0380		POP	HL
F1D7	0381			
F1D7 FB	0382		EI	
F1D8	0383			
F1D9 06 04	0384		LD	B,4 ; wait for awhile
F1DA CD BA F2	0385	WS1	CALL	DELAY
F1DD 10 FB	0386		DJNZ	WS1
F1DF	0387			
F1DF	0388			
F1DF 06 80	0389		LD	B,80H ; send the 128 NULL's...
F1E1 0E 00	0390		LD	C,0
F1E3 CD A0 F7	0391	WS2	CALL	CASOUT
F1E6 10 FB	0392		DJNZ	WS2
F1E8 0E A5	0393		LD	C,0A5H ; and the 0A5H sync byte
F1EA CD A0 F7	0394		CALL	CASOUT
F1ED 18 4C	0395		JR	SYNC4
F1EF	0396			
F1EF	0397			
F1EF	0398			SYNC - Sync on the cassette header
F1EF	0399			
F1EF	0400			
F1EF ED 73 B8 FE	0401	SYNC	LD	(MSP),SP
F1F3 31 80 FF	0402		LD	SP,ASTACK
F1F6 F3	0403		DI	
F1F7	0404			
F1F7 CD 63 F2	0405		CALL	CINIT ; do global cassette initialization
F1FA	0406			
F1FA CD BA F2	0407		CALL	DELAY ; wait for motor to speed up
F1FD	0408			
F1FD 21 8E F4	0409		LD	HL,CSTIMER-P1BIAS ; set new level 1 vector
F200 22 F2 FF	0410		LD	(INTV1),HL
F203 21 7A F4	0411		LD	HL,CSEDGE-P1BIAS ; set new level 2 vector
F206 22 F4 FF	0412		LD	(INTV2),HL
F209	0413			
F209 3A 7F FE	0414		LD	A,(SMASK) ; enable edge detector interrupt
F20C CB FF	0415		SET	7,A
F20E 32 7F FE	0416		LD	(SMASK),A
F211 D3 BE	0417		OUT	(CASPORT),A
F213	0418			
F213 FB	0419		EI	
F214	0420			

F214 06 40	0421 SYNC1 LD	B,64	; look for 64 zero bits in a row
F216 16 00	0422	LD	D,0
F218 CD 4E F2	0423 SYNC2 CALL	CRDBIT	
F21B 38 10	0424	JR	C,SYNCO
F21D 7A	0425	LD	A,D
F21E B7	0426	OR	A
F21F 20 F3	0427	JR	NZ,SYNC1
F221 10 F5	0428	DJNZ	SYNC2
F223	0429		
F223 CD 4E F2	0430 SYNC3 CALL	CRDBIT	; now start shifting in bits looking
F226 38 05	0431	JR	C,SYNCO ; for the sync byte (0A5H)
F228 7A	0432	LD	A,D
F229 FE A5	0433	CP	0A5H
F22B 20 F6	0434	JR	NZ,SYNC3
F22D	0435		
F22D F3	0436 SYNC0 DI		
F22E	0437		
F22E 21 05 F4	0438	LD	HL,CRTIMER-P1BIAS
F231 22 F2 FF	0439	LD	(INTV1),HL
F234 21 48 F4	0440	LD	HL,CEDGE-P1BIAS
F237 22 F4 FF	0441	LD	(INTV2),HL
F23A	0442		
F23A FB	0443	EI	
F23B ED 7B B8 FE	0444 SYNC4 LD	SP,(MSP)	
F23F C9	0445	RET	
F240	0446		
F240	0447		
F240	0448		
F240	0449		
F240	0450		
F240	0451		
F240	0452		
F240	0453		
F240 CD 53 F7	0454 CHKABRT CALL	KSTAT	; any key hit?
F243 B7	0455	OR	A
F244 C8	0456	RET	Z
F245 CD 58 F7	0457	CALL	KBREAD ; return if not
F248 FE 1B	0458	CP	01BH ; else set the key
F24A 37	0459	SCF	; is it the abort key?
F24B C8	0460	RET	Z ; (set carry)
F24C 3F	0461	CCF	; return if abort key
F24D C9	0462	RET	; else clear carry
F24E	0463		; and return
F24E	0464		
F24E	0465		
F24E	0466		
F24E	0467		
F24E	0468		
F24E 1E 00	0469 CRDBIT LD	E,0	; clear flag
F250	0470		
F250 CD 40 F2	0471 CRDB0 CALL	CHKABRT	; check for abort key
F253 D8	0472	RET	C ; return carry set if hit
F254 C8 7B	0473	BIT	CSAWB,E ; else did we see a clock bit yet?
F256 28 F8	0474	JR	Z,CRDB0 ; Jump if not
F259	0475		
F259 1E 00	0476	LD	E,0 ; reset saw bit flag
F25A	0477		
F25A 3E ED	0478	LD	A,-19 ; wait ~1.2 milliseconds for the data bit (if any)
F25C D3 BD	0479	OUT	(RTC),A
F25E CB 5B	0480 CRDB1 BIT	CTOUT,E	; waited enough yet?

```

F260 28 FC      0481      JR      Z,CRODB1      ; Jump if not
F262 C9         0482      RET
F263           0483
F263           0484 ;
F263           0485 ; CINIT - Do global cassette initialization
F263           0486 ;
F263           0487
F263 3A 7F FE   0488 CINIT LD      A,(SMASK)
F266 CB BF      0489      RES      7,A          ; disable edge detector interrupt
F268 CB DF      0490      SET      3,A          ; turn cassette relay on
F26A E6 CF      0491      AND      0CFH        ; set cassette state to mid
F26C 32 7F FE   0492      LD      (SMASK),A
F26F D3 BE      0493      OUT      (CASPORT),A
F271           0494
F271 AF         0495      XOR      A
F272 D3 BD      0496      OUT      (RTC),A        ; stop the realtime clock
F274 32 3D FE   0497      LD      (CRBC),A        ; clear recieve bit count
F277           0498
F277 3E 40      0499      LD      A,MCRIP
F279 32 72 FE   0500      LD      (CFLOS),A      ; clear the cassette flags
F27C           0501
F27C C9         0502      RET
F27D           0503 ;
F27D           0504 ;
F27D           0505 ; WTAIL - Write the trailer for the cassette
F27D           0506 ;
F27D           0507
F27D ED 73 B8 FE 0508 WTAIL LD      (MSP),SP
F281 31 80 FF   0509      LD      SP,ASTACK
F284 06 05      0510      LD      B,5
F286 CD BA F2   0511 WT1  CALL  DELAY
F289 10 FB      0512      DJNZ  WT1
F28B 18 07      0513      JR      RTL1
F28D           0514
F28D           0515 ;
F28D           0516 ; RTAIL - Finish up cassette read operation
F28D           0517 ;
F28D           0518
F28D ED 73 B8 FE 0519 RTAIL LD      (MSP),SP
F291 31 80 FF   0520      LD      SP,ASTACK
F294 F3         0521 RTL1  DI
F295           0522
F295 3A 7F FE   0523      LD      A,(SMASK)
F298 F6 A0      0524      OR      0A0H          ; reenale edge detector interrupt and serial control
F29A CB 9F      0525      RES      3,A          ; ...turn off the cassette motor
F29C 32 7F FE   0526      LD      (SMASK),A
F29F D3 BE      0527      OUT      (CASPORT),A
F2A1           0528
F2A1 21 60 F2   0529      LD      HL,KTIMER-P1BIAS ; reset int 1 to keyboard timer
F2A4 22 F2 FF   0530      LD      (INTV1),HL
F2A7 21 86 F1   0531      LD      HL,SEINT-P1BIAS ; and int 2 to serial edge detector
F2AA 22 F4 FF   0532      LD      (INTV2),HL
F2AD           0533
F2AD AF         0534      XOR      A
F2AE D3 BD      0535      OUT      (RTC),A        ; stop the real time clock
F2B0 32 7E FE   0536      LD      (SFLOS),A      ; clear serial flags
F2B3 32 72 FE   0537      LD      (CFLOS),A      ; clear cassette flags
F2B6           0538
F2B6 FB         0539      EI
F2B7 C3 3B F2   0540      JP      SYNC4

```

F2BA	0541
F2BA	0542 ; DELAY - Wait around for awhile
F2BA	0543
F2BA C5	0544 DELAY PUSH BC
F2BR 01 00 00	0545 LD BC,0
F2BE 0B	0546 DELY1 DEC BC
F2BF 78	0547 LD A,B
F2C0 B1	0548 OR C
F2C1 20 FB	0549 JR NZ,DELY1
F2C3 C1	0550 POP BC
F2C4 C9	0551 RET
F2C5	0552 ;
F2C5	0553 COPY SUB.CODE/1
F2C5	0554 ;
F2C5	0555 ; General subroutines
F2C5	0556 ;
F2C5	0557
F2C5	0558 ;
F2C5	0559 ; GETLINE - Get a line of data to the screen
F2C5	0560 ;
F2C5	0561
F2C5 CD 58 F7	0562 GETLINE CALL KBREAD
F2C8 FE 0D	0563 CP CCR
F2CA C8	0564 RET Z
F2CB FE 5F	0565 CP /_
F2CD 20 02	0566 JR NZ,OL1
F2CF 3E 7F	0567 LD A,07FH
F2D1 4F	0568 OL1 LD C,A
F2D2 CD DE F2	0569 CALL MPUTC
F2D5 18 EE	0570 JR GETLINE
F2D7	0571
F2D7	0572
F2D7	0573 ;
F2D7	0574 ; CRLF - Put a carriage return followed by a linefeed.
F2D7	0575 ;
F2D7	0576
F2D7	0577
F2D7 0E 0D	0578 CRLF LD C,CCR
F2D9 CD DE F2	0579 CALL MPUTC
F2DC 0E 0A	0580 LD C,CLF
F2DE	0581 ;fall thru
F2DE	0582
F2DE	0583
F2DE	0584 ;
F2DE	0585 ; MPUTC - Save current (monitor) sp, change to
F2DE	0586 ; video stack and call putchar
F2DE	0587 ;
F2DE	0588
F2DE ED 73 B8 FE	0589 MPUTC LD (MSR),SP
F2E2 31 80 FF	0590 LD SP,ASTACK
F2E5 CD 5F F3	0591 CALL PUTCHAR
F2E8 ED 7B B8 FE	0592 LD SP,(MSP)
F2EC C9	0593 RET
F2ED	0594 ;
F2ED	0595 ;
F2ED	0596 ; GETADDR - search the current screen line for
F2ED	0597 ; an address. Skip non blanks and leading spaces
F2ED	0598 ;
F2ED	0599
F2ED 06 0A	0600 GETADDR LD B,10

F2EF CD 17 F3	0601	CALL	FINDB	
F2F2 D8	0602	RET	C	
F2F3	0603	; fall thru		
F2F3	0604			
F2F3	0605			
F2F3	0606	; GHWORD - Get a hex word from the current screen line		
F2F3	0607			
F2F3	0608			
F2F3	0609			
F2F3 06 06	0610	GHWORD LD	B,6	; skip upto 6 blanks
F2F5 CD 20 F3	0611	CALL	FINDNB	
F2F8 D8	0612	RET	C	; return if too many spaces
F2F9 EB	0613	EX	DE,HL	; set screen address to de
F2FA 21 00 00	0614	LD	HL,0	; init hl
F2FD	0615			
F2FD 1A	0616	GHW1 LD	A,(DE)	; fetch the character
F2FE CD 32 F3	0617	CALL	UNHEX	; convert to hex nibble
F301 30 0B	0618	JR	NC,GHW3	; Jump if ok character
F303	0619			
F303 FE 20	0620	CP	'	; check for space
F305 28 05	0621	JR	Z,GHW2	; return with carry clear
F307 FE 3A	0622	CP	':'	; same for colon
F309 28 01	0623	JR	Z,GHW2	
F30B 37	0624	SCF		; otherwise set the carry flag
F30C EB	0625	GHW2 EX	DE,HL	
F30D C9	0626	RET		
F30E	0627			
F30E 29	0628	GHW3 ADD	HL,HL	; add in the new nibble
F30F 29	0629	ADD	HL,HL	
F310 29	0630	ADD	HL,HL	
F311 29	0631	ADD	HL,HL	
F312 85	0632	ADD	A,L	
F313 6F	0633	LD	L,A	
F314 13	0634	INC	DE	; point to the next character
F315 18 E6	0635	JR	GHW1	
F317	0636			
F317	0637			
F317	0638			
F317	0639	; FINDB - skip upto 'b' non-blanks looking for a blank		
F317	0640			
F317	0641	entry: b has maximum number of non blanks to skip		
F317	0642	hl has where to start looking		
F317	0643			
F317	0644	exit: carry set if couldn't find one		
F317	0645			
F317	0646			
F317 7E	0647	FINDB LD	A,(HL)	; set the character
F318 FE 20	0648	CP	'	; a space?
F31A C8	0649	RET	Z	; return if so
F31B 23	0650	INC	HL	; else increment pointer
F31C 10 F9	0651	DJNZ	FINDB	; decrement b and repeat if not done
F31E 37	0652	SCF		; set error flag
F31F C9	0653	RET		; and return
F320	0654			
F320	0655			
F320	0656	; FINDNB - Search for a non blank		
F320	0657			
F320	0658	entry: b has max blanks to skip		
F320	0659	hl has where to search		
F320	0660			


```

F359 C6 07      0721      ADD    A,7
F35B 4F         0722 PHN1 LD     C,A
F35C C3 DE F2   0723      JP     MPUTC
F35F            0724 ;
F35F            0725      COPY  VID.CODE/1
F35F            0726 ;
F35F            0727 ; The driver routines for the video display
F35F            0728 ;
F35F            0729 ; The video display of this machine can be looked at as two
F35F            0730 ; pages of memory.
F35F            0731 ;
F35F            0732 ; The first page contains the first 64 characters ( numbered
F35F            0733 ; 0 to 63 ) of each line ( numbered 0 to 23 ). The address of
F35F            0734 ; each character on this page can be mapped as follows:
F35F            0735 ;
F35F            0736 ;      b b b b b l l l l l c c c c c c
F35F            0737 ;
F35F            0738 ; where bbbbbb is the base address of the display ( 0F800H ),
F35F            0739 ; lllll is the line address of the character and cccccc is the
F35F            0740 ; column address of the character.
F35F            0741 ;
F35F            0742 ; The second page of the display is harder to visualize and
F35F            0743 ; explain. It contains the last 16 characters ( numbered 64 to
F35F            0744 ; 79 ) of each line. They say that one picture is worth 1000
F35F            0745 ; words so here goes.
F35F            0746 ;
F35F            0747 ;      segment 0      segment 1      segment 2
F35F            0748 ;      +-----+-----+-----+
F35F            0749 ; line 0 | FE00  line 8 | FE10  line 16 | FE20  |
F35F            0750 ;      +-----+-----+-----+
F35F            0751 ; line 1 | FE40  line 9 | FE50  line 17 | FE60  |
F35F            0752 ;      +-----+-----+-----+
F35F            0753 ; .....
F35F            0754 ;      +-----+-----+-----+
F35F            0755 ; line 6 | FF80  line 14 | FF90  line 22 | FFA0  |
F35F            0756 ;      +-----+-----+-----+
F35F            0757 ; line 7 | FF00  line 15 | FF00  line 23 | FFE0  |
F35F            0758 ;      +-----+-----+-----+
F35F            0759 ;
F35F            0760 ; Note that the segment for the next line starts 64 bytes
F35F            0761 ; from the start of the current line except when there is
F35F            0762 ; a transition from say segment 0 to segment 1. It should
F35F            0763 ; also be noted that only the first 48 bytes of each 64
F35F            0764 ; byte page ( ie 0FE00H to 0FE3FH ) are used by the video
F35F            0765 ; display. The address of a character on this page is derived
F35F            0766 ; as follows:
F35F            0767 ;
F35F            0768 ;      b b b b b b b l l l s s c c c c
F35F            0769 ;
F35F            0770 ; where bbbbbbb is the base address of the page ( 0FE00H ),
F35F            0771 ; lll is the line number the segment belongs to mod 8, ss is
F35F            0772 ; the segment the line is found in and cccc is the column
F35F            0773 ; number of the character - 64.
F35F            0774 ;
F35F            0775 ;
F35F            0776 ; a few equates before the real stuff
F35F            0777 ;
F35F            0778 ;
F35F            0779 PAGE1 EQU 0F800H      ; starting address of the first page
F35F            0780 PAGE2 EQU 0FE00H      ; starting address of the second page

```



```

F35F 0781
F35F 0782 NLines EQU 24 ; number of lines on the screen
F35F 0783 LASTLN EQU 23 ; number of the last line
F35F 0784 NGLINES EQU NLines*3 ; number of graphics lines
F35F 0785
F35F 0786 NCOLUM EQU 80 ; number of columns per line
F35F 0787 LASTCL EQU 79 ; number of the last column
F35F 0788
F35F 0789 ACURSOR EQU '_' ; an underline is our cursor
F35F 0790
F35F 0791 COLOR EQU 1 ; hi bit in SMASK if color on
F35F 0792 GRAFIX EQU 2 ; hi bit in SMASK if graphics on
F35F 0793
F35F 0794 INVERT EQU 0 ; hi bit in VFLOS if printing inverted char's
F35F 0795
F35F 0796
F35F 0797 ; define the special characters
F35F 0798
F35F 0799 CHOME EQU '^'-040H
F35F 0800 CCLEAR EQU 'Z'-040H
F35F 0801
F35F 0802 CUP EQU 'K'-040H
F35F 0803 CLEFT EQU 'H'-040H
F35F 0804 CRIGHT EQU 'L'-040H
F35F 0805
F35F 0806 CCR EQU 00DH
F35F 0807 CLF EQU 00AH
F35F 0808 CDEL EQU 07FH
F35F 0809
F35F 0810 CTAB EQU 009H
F35F 0811 CESC EQU 01BH
F35F 0812 ;
F35F 0813 ;
F35F 0814 ; PUTCHAR - write the character in c to the screen
F35F 0815 ;
F35F 0816 ; on entry:
F35F 0817 ; the character to write is in register c
F35F 0818 ; curpos and wasthere are valid
F35F 0819 ;
F35F 0820 ; on exit:
F35F 0821 ; if the character was not special (ie. cr, lf, etc.) the
F35F 0822 ; character is on the screen and in register a. If the
F35F 0823 ; character is special then the appropriate routine has
F35F 0824 ; been called. Note the these special character handlers
F35F 0825 ; should only be called from putchar
F35F 0826 ;
F35F 0827 ; trashes a
F35F 0828 ;
F35F 0829
F35F DD 22 B6 FE 0830 PUTCHAR LD (VIX),IX ; save the registers
F363 22 B4 FE 0831 LD (VHL),HL
F366 ED 53 B2 FE 0832 LD (VDE),DE
F36A ED 43 B0 FE 0833 LD (VBC),BC
F36E 0834
F36E DD 21 30 FE 0835 LD IX,VDATA
F372 2A 30 FE 0836 LD HL,(CURPOS) ; remove the cursor
F375 3A 32 FE 0837 LD A,(WASTHERE) ; put back char the cursor replaced
F378 77 0838 LD (HL),A
F379 0839
F379 3A 33 FE 0840 LD A,(ESCFLAG) ; check if processing escape sequence

```

F37C B7	0841	OR	A	
F37D CA DA F4	0842	JP	Z,PCO	
F380	0843			
F390 16 07	0844	LD	D,EESCTAB-ESCTAB/3	! escape flag is set so process
F382 CD 76 F5	0845	CALL	SWITCH	! the sequence
F385	0846			
F385 01	0847	ESCTAB DB	1	
F386 9D F3	0848	DW	QTYPE	! need type of escape sequence
F388	0849			
F388 02	0850	DB	2	
F389 2B F4	0851	DW	OLINE	! need line number for cursor address
F38B	0852			
F38B 03	0853	DB	3	
F38C 32 F4	0854	DW	OCOL	! need column number for cursor address
F38E	0855			
F38E 04	0856	DB	4	
F38F 42 F4	0857	DW	GCVAL	! need color value
F391	0858			
F391 05	0859	DB	5	
F392 5E F4	0860	DW	GOY	! get graphics bit y value
F394	0861			
F394 06	0862	DB	6	
F395 65 F4	0863	DW	GOX	! get graphics bit x value
F397	0864			
F397 07	0865	DB	7	
F398 DC F3	0866	DW	PBAI	! put character as is
F39A	0867			
F39A	0868	EESCTAB EQU	\$	
F39A	0869			
F39A C3 45 F4	0870	JP	ESDONE	
F39D	0871			
F39D	0872			! set the type of the escape sequence
F39D	0873			
F39D 79	0874	QTYPE LD	A,C	
F39E E6 7F	0875	AND	07FH	
F3A0 CD 29 F3	0876	CALL	UPSHIFT	
F3A3 16 0E	0877	LD	D,ES1TAB-S1TAB/3	! set count of state 1 table
F3A5 CD 76 F5	0878	CALL	SWITCH	
F3A8	0879			
F3A8 3D	0880	S1TAB DB	'A'	
F3A9 D4 F3	0881	DW	ISEQUAL	
F3AB	0882			
F3AB 41	0883	DB	'A'	
F3AC D8 F3	0884	DW	ISA	
F3AE	0885			
F3AE 43	0886	DB	'C'	
F3AF E3 F3	0887	DW	ISC	
F3B1	0888			
F3B1 47	0889	DB	'G'	
F3B2 FC F3	0890	DW	ISO	
F3B4	0891			
F3B4 4A	0892	DB	'J'	
F3B5 4C F4	0893	DW	ISJ	
F3B7	0894			
F3B7 4B	0895	DB	'K'	
F3B8 58 F4	0896	DW	ISK	
F3BA	0897			
F3BA 4E	0898	DB	'N'	
F3BB 02 F4	0899	DW	ISN	
F3BD	0900			

F3BD 4F	0901	DB	'O'	
F3BE EB F3	0902	DW	ISO	
F3C0	0903			
F3C0 52	0904	DB	'R'	
F3C1 08 F4	0905	DW	ISRSX	
F3C3	0906			
F3C3 53	0907	DB	'S'	
F3C4 08 F4	0908	DW	ISRSX	
F3C6	0909			
F3C6 54	0910	DB	'T'	
F3C7 0F F4	0911	DW	IST	
F3C9	0912			
F3C9 56	0913	DB	'V'	
F3CA F8 F3	0914	DW	ISV	
F3CC	0915			
F3CC 58	0916	DB	'X'	
F3CD 08 F4	0917	DW	ISRSX	
F3CF	0918			
F3CF 59	0919	DB	'Y'	
F3D0 19 F4	0920	DW	ISY	
F3D2	0921			
F3D2	0922	ES1TAB EQU	\$	
F3D2	0923			
F3D2 18 71	0924	JR	ESDONE	! abort sequence if none of the above
F3D4	0925	!		
F3D4	0926	!	Start cursor addressing sequence	
F3D4	0927			
F3D4 3E 02	0928	ISEQUAL LD	A,2	
F3D6 18 6E	0929	JR	SETESC	
F3D8	0930			
F3D8	0931	!	Put next character on screen as is	
F3D8	0932			
F3D8 3E 07	0933	ISA LD	A,7	
F3DA 18 6A	0934	JR	SETESC	
F3DC	0935			
F3DC AF	0936	PBAI XOR	A	! stop escape sequence
F3DD 32 33 FE	0937	LD	(ESCFLAQ),A	
F3E0 C3 20 F5	0938	JP	PC00	! go plot the character
F3E3	0939			
F3E3	0940	!	Turn color mode on	
F3E3	0941			
F3E3 CB 85	0942	ISC RES	0,L	! force cursor address to even byte
F3E5 DD CB 4F CE	0943	SET	COLOR,(IX+SMASK-VDATA)	! set the color on bit
F3E9 18 04	0944	JR	ISO1	
F3EB	0945			
F3ED	0946	!	Turn color mode off	
F3EB	0947			
F3EB DD CB 4F 8E	0948	ISO RES	COLOR,(IX+SMASK-VDATA)	
F3EF F3	0949	ISO1 DI		
F3F0 3A 7F FE	0950	LD	A,(SMASK)	
F3F3 D3 BE	0951	OUT	(OBEH),A	
F3F5 FB	0952	EI		
F3F6 18 4D	0953	JR	ESDONE	
F3F8	0954			
F3F8	0955	!	Set value of color byte	
F3F8	0956			
F3F8 3E 04	0957	ISV LD	A,4	
F3FA 18 4A	0958	JR	SETESC	
F3FC	0959			
F3FC	0960	!	Turn graphics mode on	

F3FC	0961				
F3FC DD CB 4F D6	0962	ISO	SET	GRAFIX,(IX+SMASK-VDATA)	; turn graphics bit on
F400 18 ED	0963	JR		ISO1	
F402	0964				
F402	0965				; Turn graphics mode off
F402	0966				
F402 DD CB 4F 96	0967	ISN	RES	GRAFIX,(IX+SMASK-VDATA)	
F406 18 E7	0968	JR		ISO1	
F408	0969				
F408	0970				; Start graphics sequence
F408	0971				
F408 32 36 FE	0972	ISRSX	LD	(OCTYPE),A	; save graphics command type
F40B 3E 05	0973		LD	A,5	; next character is Y value
F40D 18 37	0974		JR	SETESC	
F40F	0975				
F40F	0976				; Erase to end of line
F40F	0977				
F40F E5	0978	IST	PUSH	HL	
F410 CD B1 F5	0979		CALL	ATOLC	
F413 CD EA F5	0980		CALL	CLRLN	
F416 E1	0981		POP	HL	
F417 18 2C	0982		JR	ESDONE	
F419	0983				
F419	0984				; Erase to end of page
F419	0985				
F419 E5	0986	ISY	PUSH	HL	
F41A CD B1 F5	0987		CALL	ATOLC	
F41D CD EA F5	0988	ISY1	CALL	CLRLN	
F420 0E 00	0989		LD	C,0	
F422 04	0990		INC	B	
F423 78	0991		LD	A,B	
F424 FE 18	0992		CP	NLINES	
F426 20 F5	0993		JR	NZ,ISY1	
F428 E1	0994		POP	HL	
F429 18 1A	0995		JR	ESDONE	
F42B	0996				
F42B	0997				; Get line value for cursor address
F42B	0998				
F42B DD 71 05	0999	GLINE	LD	(IX+OYVAL-VDATA),C	
F42E 3E 03	1000		LD	A,3	
F430 18 14	1001		JR	SETESC	
F432	1002				
F432	1003				; Get column value for cursor address and set
F432	1004				
F432 11 18 1F	1005	OCOL	LD	DE,31*256+NLINES	; load de with mask and max value
F435 CD BE F4	1006		CALL	CHKLINE	
F439 47	1007		LD	B,A	; save new line number
F439 CD C8 F4	1008		CALL	CHKCOL	; check for valid column number
F43C 4F	1009		LD	C,A	
F43D CD 88 F5	1010		CALL	LCTOA	
F440 18 03	1011		JR	ESDONE	; done with escape sequence
F442	1012				
F442	1013				; Get color byte value
F442	1014				
F442 DD 71 04	1015	GCVAL	LD	(IX+CVAL-VDATA),C	; set here to set the color value
F445	1016	JR	ESDONE		; fall thru
F445	1017				
F445	1018				
F445 AF	1019	ESDONE	XOR	A	; set here when done with sequence or error
F446 32 33 FE	1020	SETESC	LD	(ESCFLAG),A	; set new sequence value

F449 C3 53 F5	1021	JP	PUTC3	
F44C	1022			
F44C	1023			; Start inverted video
F44C	1024			
F44C DD CB 4F 56	1025	ISJ	BIT	GRAFIX,(IX+SMASK-VDATA) ; don't do it if in graphics
F450 20 F3	1026	JR		NZ,ESDONE
F452 DD CB 07 C6	1027	SET		INVERT,(IX+VFLGS-VDATA)
F454 18 ED	1028	JR		ESDONE
F458	1029			
F458	1030			; Stop inverted video
F458	1031			
F458 DD CB 07 86	1032	ISK	RES	INVERT,(IX+VFLGS-VDATA)
F45C 18 E7	1033	JR		ESDONE
F45E	1034			
F45E	1035			; Get y value for graphics stuff
F45E	1036			
F45E DD 71 05	1037	GOY	LD	(IX+OYVAL-VDATA),C ; save Y value of bit
F461 3E 06	1038	LD		A,6 ; next state is set X
F463 18 E1	1039	JR		SETESC
F465	1040			
F465	1041			; Get x value for graphics stuff andiddle the bit
F465	1042			
F465 DD CB 4F 56	1043	GOX	BIT	GRAFIX,(IX+SMASK-VDATA) ; graphics mode on?
F469 28 DA	1044	JR		Z,ESDONE ; abort if not
F46B 11 48 7F	1045	LD		DE,127*256+NGLINES ; load mask and max value
F46E CD BE F4	1046	CALL		CHKLINE
F471	1047			
F471	1048			; compute line number of cell for bit
F471	1049			
F471 06 FF	1050	GOX0	LD	B,-1 ; init line counter
F473 04	1051	GOX1	INC	B ; increment line counter
F474 D6 03	1052	SUB		A,3 ; cheap divide and modulo by 3
F476 30 FB	1053	JR		NC,GOX1
F478 C6 03	1054	ADD		A,3 ; set mod, it's the bit number >> 1
F47A 5F	1055	LD		E,A ; save in e, b has line number
F47B	1056			
F47B	1057			; compute column number of cell for bit
F47B	1058			
F47B 79	1059	LD		A,C ; set biased X value
F47C D6 20	1060	SUB		A, ; remove bias
F47E 1F	1061	RRA		; divide by 2 and even/odd bit to carry
F47F CB 13	1062	RL		E ; rotate even/odd bit into bit number
F481	1063			
F481 CD CB F4	1064	CALL		CC1 ; check for valid column
F484	1065			
F484 4F	1066	LD		C,A ; set column address
F485 E5	1067	PUSH		HL ; save current screen address
F486 CD 88 F5	1068	CALL		LCTOA ; set screen address of cell for bit
F489	1069			
F489 7E	1070	LD		A,(HL) ; check if not graphic cell yet
F48A 17	1071	RLA		; set MSB to carry
F48B 38 02	1072	JR		C,GOX4 ; Jump if graphics cell
F48D 36 80	1073	LD		(HL),080H ; otherwise make it one
F48F	1074			
F48F 3E 80	1075	GOX4	LD	A,080H ; set a bit in the most sig. bit
F491 43	1076	LD		B,E ; set bit number to b
F492 04	1077	INC		B
F493 07	1078	GOX5	RLCA	; rotate bit to next position left
F494 10 FD	1079	DJNZ		GOX5 ; decrement bit number, Jump if not zero
F496 5F	1080	LD		E,A ; save mask in e


```

F497      1081
F497 3A 36 FE 1082      LD      A,(OCTYPE)      ; set graphics command type ( R,S or X)
F49A FE 53   1083      CP      'S'              ; check for set
F49C 7B      1084      LD      A,E              ; (set mask)
F49D 38 08   1085      JR      C,ISR            ; Jump if command is R
F49F 20 03   1086      JR      NZ,ISX          ; Jump if command is X
F4A1      1087
F4A1 B6      1088      OR      (HL)              ; command is set, set the bit
F4A2 18 05   1089      JR      CHKMTY          ; so check if cell is empty
F4A4      1090 ;
F4A4 AE      1091 ISX   XOR      (HL)              ; command is exclusive-or, flip the bit
F4A5 18 02   1092      JR      CHKMTY
F4A7      1093
F4A7 2F      1094 ISR   CPL                      ; command is reset, complement the mask...
F4A8 A6      1095      AND      (HL)              ; ...and reset the bit
F4A9      1096
F4A9 77      1097 CHKMTY LD      (HL),A          ; restore the cell
F4AA E6 7F   1098      AND      07FH              ; check if the cell is empty
F4AC 20 02   1099      JR      NZ,G0X6          ; Jump if not
F4AE 36 20   1100      LD      (HL),              ; else make the cell a space
F4B0 DD CB 4F 4E 1101 G0X6 BIT      COLOR,(IX+SMASK-VDATA)
F4B4 28 05   1102      JR      Z,G0X7
F4B6 23      1103      INC      HL
F4B7 3A 34 FE 1104      LD      A,(CVAL)
F4BA 77      1105      LD      (HL),A
F4BB E1      1106 G0X7 POP      HL              ; restore screen address of the cursor
F4BC 18 87   1107      JR      ESDONE
F4BE      1108
F4BE      1109 ;
F4BE      1110 ; CHKLINE - Check biased line number in GYVAL for validity
F4BE      1111 ; return valid line in A
F4BE      1112 ;
F4BE      1113 ; D has mask value, E has max line value
F4BE      1114 ;
F4BE      1115
F4BE 3A 35 FE 1116 CHKLINE LD      A,(GYVAL)
F4C1 D6 20   1117      SUB      A,              ; remove bias
F4C3 A2      1118      AND      D              ; keep it legal
F4C4 BB      1119      CP      E              ; check for legal value
F4C5 D8      1120      RET      C              ; return if legal
F4C6 93      1121      SUB      A,E              ; else make it legal
F4C7 C9      1122      RET
F4C8      1123
F4C8      1124 ;
F4C8      1125 ;
F4C8      1126 ; CHKCOL - Check biased column number in C for validity
F4C8      1127 ; Return valid column in A
F4C8      1128 ;
F4C8      1129 ;
F4C8 79      1130 CHKCOL LD      A,C              ; set the column address
F4C9 D6 20   1131      SUB      A,              ; remove ascii bias
F4CB DD CB 4F 4E 1132 CC1 BIT      COLOR,(IX+SMASK-VDATA) ; check if in color mode
F4CF 28 01   1133      JR      Z,CC2              ; Jump if not
F4D1 87      1134      ADD      A,A              ; else adjust column address
F4D2 E6 7F   1135 CC2 AND      127
F4D4 FE 50   1136      CP      NCOLUM
F4D6 D8      1137      RET      C
F4D7 D6 50   1138      SUB      A,NCOLUM
F4D9 C9      1139      RET
F4DA      1140

```



```

F4DA      1141 ; end of escape sequence stuff
F4DA      1142 ;
F4DA      1143 ; Get here when not in escape sequence
F4DA      1144
F4DA 79    1145 PC0   LD     A,C
F4DB E6 7F 1146      AND    07FH      ; keep the hi bit down
F4DD 4F    1147      LD     C,A
F4DE 28 73 1148      JR     Z,PUTC3
F4E0 FE 7F 1149      CP     CDEL
F4E2 28 04 1150      JR     Z,PC1
F4E4 FE 20 1151      CP     '
F4E6 30 2A 1152      JR     NC,PUTC      ; all other special characters are < space
F4E8      1153
F4E8 F5    1154 PC1   PUSH  AF      ; most of the routines for the special
F4E9 CD B1 F5 1155      CALL  ATOLC      ; characters need line/column info,
F4EC F1      1156      POP   AF      ; so get it here
F4ED      1157
F4ED 16 0A 1158      LD     D,ECHRTAB-CHRTAB/3
F4EF CD 76 F5 1159      CALL  SWITCH
F4F2      1160
F4F2 0D    1161 CHRTAB DB  CCR
F4F3 9B F6 1162      DW     CR
F4F5      1163
F4F5 0A    1164      DB     CLF
F4F6 90 F6 1165      DW     LF
F4F8      1166
F4F8 7F    1167      DB     CDEL
F4F9 C2 F6 1168      DW     DEL
F4FB      1169
F4FB 0B    1170      DB     CUP
F4FC 5E F6 1171      DW     UP
F4FE      1172
F4FE 0B    1173      DB     CLEFT
F4FF 67 F6 1174      DW     LEFT
F501      1175
F501 0C    1176      DB     CRIGHT
F502 79 F6 1177      DW     RIGHT
F504      1178
F504 1E    1179      DB     CHOME
F505 22 F6 1180      DW     HOME
F507      1181
F507 1A    1182      DB     CCLEAR
F508 DE F5 1183      DW     CLEAR
F50A      1184
F50A 09    1185      DB     CTAB
F50B A0 F6 1186      DW     TAB
F50D      1187
F50D 1B    1188      DB     CESC
F50E D7 F5 1189      DW     ESC
F510      1190
F510      1191 ECHRTAB EQU $      ; end of the table
F510      1192
F510 18 3E 1193      JR     PUTC2      ; any characters < ' ' and not in table are ignored
F512      1194 ;
F512      1195 ; Get here if character not special
F512      1196
F512 DD CB 07 46 1197 PUTC BIT  INVERT, (IX+VFLOS-VDATA) ; check for inverted video
F516 28 08 1198      JR     Z,PC00      ; Jump if not in mode
F518 DD CB 4F 56 1199      BIT  GRAFIX, (IX+SMASK-VDATA) ; check if in graphics mode
F51C 20 02 1200      JR     NZ,PC00      ; if so then don't set invert bit

```

F51E		1201			
F51E CB F9		1202	SET	7,C	; set the invert bit
F520		1203			
F520 71		1204	PC00 LD	(HL),C	; char was not special, place it on the screen
F521 23		1205	INC	HL	; point to next loc of screen
F522 DD CB 4F 4E		1206	BIT	COLOR,(IX+SMASK-VDATA)	; check if in color mode
F526 28 05		1207	JR	Z,PC01	; Jump if not
F528		1208			
F528 3A 34 FE		1209	LD	A,(CVAL)	; else next byte sets the color value,
F52B 77		1210	LD	(HL),A	; set the color
F52C 23		1211	INC	HL	
F52D		1212			
F52D 7D		1213	PC01 LD	A,L	; check for page crossing
F52E E6 3F		1214	AND	03FH	
F530 20 09		1215	JR	NZ;PUTC1	; Jump if no crossing
F532		1216			
F532 2B		1217	DEC	HL	
F533 CD B1 F5		1218	CALL	ATOLC	; else set line/column
F536 0C		1219	INC	C	; set to next column
F537 C3 50 F5		1220	JP	PUTC2	
F53A		1221			
F53A E6 0F		1222	PUTC1 AND	00FH	; check for possible end of line
F53C 20 15		1223	JR	NZ,PUTC3	
F53E 7C		1224	LD	A,H	
F53F FE FE		1225	CP	PAGE2/256	; check if off bottom of screen
F541 38 10		1226	JR	C,PUTC3	; Jump if not
F543		1227			
F543 2B		1228	DEC	HL	
F544 CD B1 F5		1229	CALL	ATOLC	
F547 78		1230	LD	A,B	
F548 04		1231	INC	B	
F549 FE 17		1232	CP	LASTLN	
F54B D4 28 F6		1233	CALL	NC,SCROLL	
F54E 0E 00		1234	LD	C,0	
F550 CD 88 F5		1235	PUTC2 CALL	LCTOA	
F553		1236			
F553 7E		1237	PUTC3 LD	A,(HL)	; set character cursor will cover
F554 32 32 FE		1238	LD	(WASTHERE),A	; save it
F557 CB FF		1239	SET	7,A	; set hi bit for cursor
F559		1240			
F559 DD CB 4F 56		1241	BIT	GRAFIX,(IX+SMASK-VDATA)	; check for graphics mode
F55D 28 02		1242	JR	Z,PUTC4	; Jump if not in graphics mode
F55F		1243			
F55F 3E 5F		1244	LD	A,ACURSOR	; when in graphics mode use underline
F561		1245			
F561 77		1246	PUTC4 LD	(HL),A	; in goes the cursor
F562 22 30 FE		1247	LD	(CURPOS),HL	; save current cursor address
F565 ED 4B B0 FE		1248	LD	BC,(VBC)	
F569 ED 5B B2 FE		1249	LD	DE,(VDE)	
F56D 2A B4 FE		1250	LD	HL,(VHL)	
F570 DD 2A B6 FE		1251	LD	IX,(VIX)	
F574 79		1252	LD	A,C	; return character in a and c
F575 C9		1253	RET		
F576		1254			
F576		1255			
F576		1256	SWITCH	- scan a table pointed to by the address on the top	
F576		1257		of the stack. The character to match is in register a	
F576		1258			
F576		1259		on entry!	
F576		1260		register a contains the character to find	


```

F576      1261 ;      register d contains the number of entries in the table
F576      1262 ;
F576      1263 ;      on exit:
F576      1264 ;      if a match is found then vector the the associated
F576      1265 ;      address else return to the location following
F576      1266 ;      the table
F576      1267 ;
F576      1268 ;      trashes d
F576      1269 ;
F576      1270
F576 E3    1271 SWITCH EX (SP),HL ; set address of the table
F577 BE    1272 SW1 CP (HL) ; check for match
F578 23    1273 INC HL ; increment pointer
F579 28 07 1274 JR Z,SW2 ; Jump if match
F57B 23    1275 INC HL ; skip address part
F57C 23    1276 INC HL
F57D 15    1277 DEC D ; decrement counter
F57E 28 06 1278 JR Z,SW3 ; Jump if end of table
F580 18 F5 1279 JR SW1 ; and try again
F582      1280
F582 56    1281 SW2 LD D,(HL) ; set low byte of address
F583 23    1282 INC HL
F584 66    1283 LD H,(HL) ; set hi byte
F585 6A    1284 LD L,D
F586 E3    1285 SW3 EX (SP),HL
F587 C9    1286 RET
F588      1287 ;
F588      1288 ;
F588      1289 ; LCTOA - convert the line/column relative address in bc to the
F588      1290 ; actual memory address of the character. Note that no
F588      1291 ; range checking is done, this routine must be passed
F588      1292 ; valid data.
F588      1293 ;
F588      1294 ; on entry:
F588      1295 ; b contains the line number of the character
F588      1296 ; c contains the column number of the character
F588      1297 ;
F588      1298 ; on exit:
F588      1299 ; hl contains the memory address of the character
F588      1300 ;
F588      1301 ; trashes d,hl
F588      1302 ;
F588      1303
F588 79    1304 LCTOA LD A,C ; set the column address
F589 FE 40 1305 CP 64 ; check if in tail end
F58B 38 16 1306 JR C,LCTA1
F58D      1307
F58D      1308 ; set here when column is greater than 63
F58D      1309
F58D E6 0F 1310 AND 00FH ; strip out the least sig four bits ( 0 - 15 )
F58F 6F    1311 LD L,A ; they form the 4 least sig bits of the address
F590 78    1312 LD A,B ; set line number
F591 0F    1313 RRCA ; shift out 2 lsb's of line
F592 0F    1314 RRCA
F593 57    1315 LD D,A
F594 F6 FE 1316 OR PAGE2/256 ; or' in base address
F595 F6 FE 1317 OR PAGE3/256 ; or' in hi byte of address
F596 67    1318 LD A,D
F597 7A    1319 AND 000H ; leave 2 lsb's of line
F598 E6 C0 1320 OR L ; or' in column
F59A B5

```


F59B 6F	1321	LD	L,A	
F59C 78	1322	LD	A,B	
F59D 07	1323	RLCA		
F59E E6 30	1324	AND	030H	; shift segment bits left one
F5A0 B5	1325	OR	L	; Just leave the seg bits
F5A1 6F	1326	LD	L,A	; or to 10 byte of address
F5A2 C9	1327	RET		
F5A3	1328			
F5A3	1329	; set here when column is < 64		
F5A3	1330			
F5A3 6F	1331	LCTA1 LD	L,A	; column goes to 10 byte of address
F5A4 78	1332	LD	A,B	; set line
F5A5 0F	1333	RRCA		
F5A6 0F	1334	RRCA		; must split the line number between hi and lo
F5A7 57	1335	LD	D,A	
F5A8 E6 C0	1336	AND	0COH	; leave 2 lsb's of line
F5AA B5	1337	OR	L	; or with column
F5AB 6F	1338	LD	L,A	; to form 10 byte of address
F5AC 7A	1339	LD	A,D	
F5AD F6 F8	1340	OR	PAGE1/256	; or in base address of page 1
F5AF 67	1341	LD	H,A	; to hi byte of address
F5B0 C9	1342	RET		
F5B1	1343			
F5B1	1344			
F5B1	1345	ATOLC - convert an address to the equivalent line/column		
F5B1	1346	address. The same note as in LCTOA applies about range		
F5B1	1347	checking		
F5B1	1348			
F5B1	1349	on entry:		
F5B1	1350	hl contains the address to convert		
F5B1	1351			
F5B1	1352	on exit:		
F5B1	1353	b contains the line number		
F5B1	1354	c contains the column number		
F5B1	1355			
F5B1	1356	trashes bc,de		
F5B1	1357			
F5B1	1358			
F5B1 54	1359	ATOLC LD	D,H	
F5B2 5D	1360	LD	E,L	
F5B3 7C	1361	LD	A,H	; check for page one
F5B4 FE FE	1362	CP	PAGE2/256	
F5B6 7D	1363	LD	A,L	; always start with column (the easy one)
F5B7 30 0B	1364	JR	NC,ATOL1	; Jump if page two
F5B9	1365			
F5B9 E6 3F	1366	AND	63	; strip off 2 lsb's of line number
F5BB 4F	1367	LD	C,A	
F5BC 29	1368	ADD	HL,HL	; shift hl 2 bits left, line ends up in h
F5BD 29	1369	ADD	HL,HL	
F5BE 7C	1370	LD	A,H	; set line
F5BF E6 1F	1371	AND	31	; strip base address
F5C1 47	1372	LD	B,A	
F5C2 EB	1373	EX	DE,HL	
F5C3 C9	1374	RET		
F5C4	1375			
F5C4	1376	; set here when in page two		
F5C4	1377			
F5C4 E6 0F	1378	ATOL1 AND	00FH	; leave 4 lsb's of column
F5C6 C6 40	1379	ADD	A,64	; must be column 64 - 79 if in page two
F5C8 4F	1380	LD	C,A	

F5C9 7D	1381	LD	A,L	
F5CA 0F	1382	RRCA		; Justify the segment bits
F5CB E6 18	1383	AND	018H	; remove everything else
F5CD 47	1384	LD	B,A	
F5CE 29	1385	ADD	HL,HL	; shift all line number bits to h
F5CF 29	1386	ADD	HL,HL	
F5D0 7C	1387	LD	A,H	
F5D1 E6 07	1388	AND	007H	; leave the line number bits
F5D3 80	1389	OR	B	; or'in seg bits to form full line number
F5D4 47	1390	LD	B,A	
F5D5 EB	1391	EX	DE,HL	
F5D6 C9	1392	RET		
F5D7	1393			
F5D7	1394			
F5D7	1395 ;			
F5D7	1396 ;	ESC - Start an escape sequence		
F5D7	1397 ;			
F5D7	1398			
F5D7 DD 36 03 01	1399	ESC LD	(IX+ESCFLAG-VDATA),1	; set state to need second char
F5D8 C3 50 F5	1400	JP	PUTC2	
F5DE	1401 ;			
F5DE	1402 ;			
F5DE	1403 ;	CLEAR - clear the video memory. Note that this routine relies		
F5DE	1404 ;	on the fact that the video memory effectively ends at		
F5DE	1405 ;	OFFFFH.		
F5DE	1406 ;			
F5DE	1407 ;	on entry:		
F5DE	1408 ;	who cares		
F5DE	1409 ;			
F5DE	1410 ;	on exit:		
F5DE	1411 ;	the screens been cleared		
F5DE	1412 ;			
F5DE	1413 ;	trashes a,b,h		
F5DE	1414 ;			
F5DE	1415			
F5DE 01 00 17	1416	CLEAR LD	BC, LASTLN*256+0	; set line/column to 23/0
F5E1 CD EA F5	1417	CLR1 CALL	CLRLN	; clear a line
F5E4 05	1418	DEC	B	; decrement line number
F5E5 F2 E1 F5	1419	JP	P,CLR1	; Jump if not
F5E8 18 38	1420	JR	HOME	; now home the cursor
F5EA	1421			
F5EA	1422			
F5EA	1423 ;			
F5EA	1424 ;	CLRLN - Clear a video line starting at the current		
F5EA	1425 ;	column value		
F5EA	1426 ;			
F5EA	1427 ;	entry: bc has line/column		
F5EA	1428 ;			
F5EA	1429 ;	exit: the line has been cleared		
F5EA	1430 ;			
F5EA	1431			
F5EA C5	1432	CLRLN PUSH	BC	; save bc
F5EB CD 88 F5	1433	CALL	LCTOA	; convert to address
F5EE 3E 3F	1434	LD	A,63	; compute positions to clear in first part of line
F5F0 91	1435	SUB	A,C	
F5F1 38 07	1436	JR	C,CLN1	; Jump if none
F5F3 3C	1437	INC	A	; adjust count
F5F4 4F	1438	LD	C,A	; move count to c
F5F5 CD 06 F6	1439	CALL	CLRMEM	; clear the memory
F5F8 0E 40	1440	LD	C,64	; start column of second part of line

F5FA CD 88 F5	1441 CLN1	CALL	LCTOA	; convert to address
F5FD 3E 50	1442	LD	A,NCOLUM	; compute positions to clear
F5FF 91	1443	SUB	A,C	
F600 4F	1444	LD	C,A	; c gets count
F601 CD 06 F6	1445	CALL	CLRMEM	; clear tail end of the line
F604 C1	1446 CLN2	POP	BC	; restore line/column
F605 C9	1447	RET		
F606	1448 ;			
F606	1449 ;			
F606	1450 ;	CLRMEM	- Clear a block of memory. If we're in color mode	
F606	1451 ;		then set the color bytes to the current value	
F606	1452 ;			
F606	1453 ;		on entry:	
F606	1454 ;		hl points to the block	
F606	1455 ;		c has the size of the block	
F606	1456 ;			
F606	1457 ;		on exit:	
F606	1458 ;		the block has been cleared	
F606	1459 ;			
F606	1460 ;		trashes a,c,d,e,hl	
F606	1461 ;			
F606	1462			
F606 54	1463 CLRMEM LD	D,H		; make copy of hl in de
F607 5D	1464	LD	E,L	
F608	1465			
F608 36 20	1466	LD	(HL),C	; zap first character
F60A 23	1467	INC	HL	; increment pointer
F60B 0D	1468	DEC	C	; decrement count
F60C C8	1469	RET	Z	
F60D DD CB 4F 4E	1470	BIT	COLOR,(IX+SMASK-VDATA)	; check if in color mode
F611 28 07	1471	JR	Z,CLRM1	; Jump if not
F613	1472			
F613 3A 34 FE	1473	LD	A,(CVAL)	; set next byte as color value
F616 77	1474	LD	(HL),A	
F617 23	1475	INC	HL	; increment pointer
F618 0D	1476	DEC	C	; decrement count
F619 C8	1477	RET	Z	
F61A	1478			
F61A 1A	1479 CLRM1 LD	A,(DE)		
F61B 77	1480	LD	(HL),A	
F61C 13	1481	INC	DE	
F61D 23	1482	INC	HL	
F61E 0D	1483	DEC	C	
F61F 20 F9	1484	JR	NZ,CLRM1	
F621 C9	1485	RET		
F622	1486			
F622	1487			
F622	1488 ;			
F622	1489 ;	HOME	- position the cursor to the upper left corner of the	
F622	1490 ;		screen.	
F622	1491 ;			
F622	1492 ;		on entry:	
F622	1493 ;		curpos contains the current address of the cursor	
F622	1494 ;		wasthere contains the char which was were the cursor was	
F622	1495 ;			
F622	1496 ;		on exit:	
F622	1497 ;		the cursor is in the upper left corner	
F622	1498 ;		curpos contains the new address of the cursor	
F622	1499 ;		wasthere contains the char which was in the upper left	
F622	1500 ;			

F622		1501 ;	trashes hl	
F622		1502 ;		
F622		1503		
F622	21 00 F8	1504 HOME LD	HL,PAGE1	; address of the upper left corner
F625	C3 53 F5	1505 JP	PUTC3	
F628		1506 ;		
F628		1507 ;		
F628		1508 ;	SCROLL - scroll the screen up one line and clear the bottom	
F628		1509 ;	line (line 23).	
F628		1510 ;		
F628		1511 ;	on entry:	
F628		1512 ;	who cares	
F628		1513 ;		
F628		1514 ;	on exit:	
F628		1515 ;	the screen has been scrolled up one line and the bottom	
F628		1516 ;	line has been cleared	
F628		1517 ;		
F628		1518 ;	trashes everything	
F628		1519 ;		
F628		1520		
F628	11 00 F8	1521 SCROLL LD	DE,PAGE1	; point to start of Page 1
F628	D5	1522 PUSH	DE	; save it on the stack
F62C	11 00 FE	1523 LD	DE,PAGE2	; point to start of Page 2
F62F	06 00	1524 LD	B,0	; initialize hi byte of count
F631		1525		
F631	21 40 00	1526 SC1 LD	HL,64	; de has destination address.....
F634	19	1527 ADD	HL,DE	; ...compute source address
F635	30 04	1528 JR	NC,SC2	; Jump if no overflow
F637	21 50 FE	1529 LD	HL,64+PAGE2+16	; else compute new source address
F63A	19	1530 ADD	HL,DE	
F63B	E5	1531 SC2 PUSH	HL	; save source (next destination)
F63C	0E 10	1532 LD	C,16	; set move count
F63E	ED B0	1533 LDIR		; do the move
F640	E1	1534 POP	HL	; set next page 2 destination address
F641	D1	1535 POP	DE	; set page 1 destination address
F642	E5	1536 PUSH	HL	; save page 2 destination address
F643		1537		
F643	21 40 00	1538 LD	HL,64	; compute page 1 source address
F646	19	1539 ADD	HL,DE	
F647	3E 04	1540 LD	A,4	; move line in 4 segments
F649	0E 10	1541 SC3 LD	C,16	; move a segment
F64B	ED B0	1542 LDIR		
F64D	3D	1543 DEC	A	; check for more
F64E	20 F9	1544 JR	NZ,SC3	; Jump if so
F650		1545		
F650	7C	1546 LD	A,H	
F651	EB	1547 EX	DE,HL	
F652	D1	1548 POP	DE	; else set page 2 destination
F653	E5	1549 PUSH	HL	; save next page 1 destination
F654	FE FE	1550 CP	PAGE2/256	
F656	20 D9	1551 JR	NZ,SC1	
F658	E1	1552 POP	HL	
F659		1553		
F659	01 00 17	1554 LD	BC,LASTLN*256+0	
F65C	18 BC	1555 JR	CLRLN	
F65E		1556 ;		
F65E		1557 ;		
F65E		1558 ;	UP - move the cursor up one line. If the cursor is currently	
F65E		1559 ;	on line 0 then the cursor wraps around to line 23	
F65E		1560 ;		

```

F65E      1561 ;      on entry:
F65E      1562 ;      bc contains the current line/column address
F65E      1563 ;
F65E      1564 ;      on exit:
F65E      1565 ;      the cursor is up one line
F65E      1566 ;
F65E      1567 ;      trashes a,b,c,hl
F65E      1568 ;
F65E      1569 ;
F65E 05    1570 UP    DEC    B
F65F F2 50 F5 1571    JP     P,PUTC2      ; check for wrap around
F662 06 17    1572    LD     B,LASTLN    ; go to bottom if it went off the top
F664 C3 50 F5 1573    JP     PUTC2      ; finish up
F667      1574
F667      1575
F667      1576 ;
F667      1577 ; LEFT - move the cursor one character position to the left.
F667      1578 ; If it falls off the left edge, put it on the right
F667      1579 ;
F667      1580 ;      on entry:
F667      1581 ;      bc has the current line column address
F667      1582 ;
F667      1583 ;      on exit:
F667      1584 ;      it's been done
F667      1585 ;
F667      1586 ;      trashes a,b,c,hl
F667      1587 ;
F667      1588 ;
F667 DD CB 4F 4E 1589 LEFT BIT    COLOR,(IX+SMASK-VDATA) ; check if in color mode
F66B 28 01      1590    JR     Z,LFT1    ; Jump if not
F66D 0D          1591    DEC    C        ; decrement column once for color mode
F66E 0D          1592 LFT1 DEC    C
F66F F2 50 F5   1593    JP     P,PUTC2      ; Jump if column still positive
F672 3E 50      1594    LD     A,NCOLUM    ; else compute new column number
F674 81          1595    ADD    A,C
F675 4F          1596    LD     C,A
F676 C3 50 F5   1597    JP     PUTC2
F679      1598
F679      1599
F679      1600 ;
F679      1601 ; RIGHT - move the cursor right with wrap around
F679      1602 ;
F679      1603 ;      on entry:
F679      1604 ;      bc contains the current line/column address
F679      1605 ;
F679      1606 ;
F679      1607 ;
F679 DD CB 4F 4E 1608 RIGHT BIT    COLOR,(IX+SMASK-VDATA)
F67D 28 01      1609    JR     Z,RT1
F67F 0C          1610    INC    C
F680 0C          1611 RT1   INC    C
F681 79          1612    LD     A,C
F682 FE 50      1613    CP     NCOLUM
F684 38 02      1614    JR     C,RT2
F686 0E 00      1615    LD     C,0
F688 C3 50 F5   1616 RT2   JP     PUTC2
F68B      1617 ;
F68B      1618 ;
F68B      1619 ; CR - process a carriage return
F68B      1620 ;

```



```

F68B      1621 ; on entry:
F68B      1622 ; bc has line column
F68B      1623 ;
F68B      1624 ; on exit:
F68B      1625 ; the cursor is in column zero
F68B      1626 ;
F68B      1627 ; trashes a,b,c,hI
F68B      1628 ;
F68B      1629 ;
F68B OE 00 1630 OR LD C,0 ; set column to zero
F68D C3 50 F5 1631 JP PUTC2
F690      1632
F690      1633
F690      1634 ;
F690      1635 ; LF - Process a linefeed
F690      1636 ;
F690      1637 ; on entry:
F690      1638 ; bc has line/column
F690      1639 ;
F690      1640 ; on exit:
F690      1641 ; bc has new line column address. If cursor went off
F690      1642 ; bottom of screen, the screen was scrolled
F690      1643 ;
F690      1644 ; trashes a,b,c,d,e,hI
F690      1645 ;
F690      1646 ;
F690 78      1647 LF LD A,B
F691 04      1648 INC B
F692 FE 17   1649 CP LASTLN
F694 38 07   1650 JR C,LF1
F696 C5      1651 PUSH BC ; save column address
F697 CD 28 F6 1652 CALL SCROLL
F69A C1      1653 POP BC
F69B 06 17   1654 LD B, LASTLN
F69D C3 50 F5 1655 LF1 JP PUTC2
F6A0      1656 ;
F6A0      1657 ;
F6A0      1658 ; TAB - Process the tab character
F6A0      1659 ;
F6A0      1660 ; on entry:
F6A0      1661 ; bc has current line/column
F6A0      1662 ;
F6A0      1663 ; on exit:
F6A0      1664 ; the cursor is pointed at the next tab stop
F6A0      1665 ;
F6A0      1666 ; trashes a,b,c,hI
F6A0      1667 ;
F6A0      1668 ;
F6A0 79      1669 TAB LD A,C ; set column address
F6A1 DD CB 4F 4E 1670 BIT COLOR, (IX+SMASK-VDATA) ; check for color mode
F6A5 20 06   1671 JR NZ, TAB1 ; Jump if in color mode
F6A7      1672
F6A7 C6 08   1673 ADD A, 8 ; set column to next multiple of eight - 1
F6A9 E6 F8   1674 AND -8
F6AB 18 04   1675 JR TAB2
F6AD      1676
F6AD C6 10   1677 TAB1 ADD A, 16 ; set column to next multiple of sixteen - 1
F6AF E6 F0   1678 AND -16
F6B1      1679
F6B1 4F      1680 TAB2 LD C, A

```



```

F6B2 FE 50      1681      CP      NCOLUM      ; check if past end of line
F6B4 38 09      1682      JR      C,TAB3      ; Jump if not
F6B6            1683
F6B6 0E 00      1684      LD      C,0          ; else set column to zero
F6B8 78         1685      LD      A,B          ; ...and increment line number
F6B9 04         1686      INC      B
F6BA FE 17      1687      CP      LASTLN      ; check if on last line
F6BC D4 28 F6   1688      CALL    NC,SCROLL    ; call if so
F6BF C3 50 F5   1689 TAB3  JP      PUTC2
F6C2            1690
F6C2            1691
F6C2            1692 ;
F6C2            1693 ; DEL - process the delete characters
F6C2            1694 ;
F6C2            1695 ; on entry:
F6C2            1696 ; bc has current line/column
F6C2            1697 ;
F6C2            1698 ; on exit:
F6C2            1699 ; the previous character has been deleted
F6C2            1700 ;
F6C2            1701 ; trashes a,b,c,hI
F6C2            1702 ;
F6C2            1703
F6C2 DD CB 4F 4E 1704 DEL  BIT      COLOR,(IX+SMASK-VDATA)
F6C6 28 01      1705      JR      Z,DELO
F6C8 0D         1706      DEC      C
F6C9 0D         1707 DELO  DEC      C
F6CA F2 D7 F6   1708      JP      P,DEL1
F6CD 3E 50      1709      LD      A,NCOLUM
F6CF 81         1710      ADD     A,C
F6D0 4F         1711      LD      C,A
F6D1 05         1712      DEC     B
F6D2 F2 D7 F6   1713      JP      P,DEL1
F6D5 06 17      1714      LD      B,LASTLN
F6D7 CD 88 F5   1715 DEL1  CALL    LCTOA
F6DA 36 20      1716      LD      (HL),
F6DC C3 53 F5   1717      JP      PUTC3
F6DF            1718 ;
F6DF            1719      COPY   SER.CODE/1
F6DF            1720 ;
F6DF            1721 ; SELBAUD - Monitor command entry for selecting serial
F6DF            1722 ; baud rate
F6DF            1723 ;
F6DF            1724 ; commadn syntax: b <0-1>
F6DF            1725 ;
F6DF            1726
F6DF CD ED F2   1727 SELBAUD CALL GETADDR      ; set the flag
F6E2 DA 96 F0   1728      JP      C,ERROR      ; Jump if none
F6E5 7B         1729      LD      A,E          ; move it into A
F6E6 CD EC F6   1730      CALL    SERSEL      ; select the value
F6E9 C3 98 F0   1731      JP      NXTCMD      ; set next command
F6EC            1732
F6EC            1733
F6EC            1734 ;
F6EC            1735 ; SERSEL - select the serial baud rate
F6EC            1736 ;
F6EC            1737 ; usage: call sersel
F6EC            1738 ;
F6EC            1739 ; entry: a contains the baud rate selector
F6EC            1740 ; 0 = 300 baud

```

```

F6EC      1741 ;      1 = 1200 baud
F6EC      1742 ;
F6EC      1743 ; exit: the rate has been selected
F6EC      1744 ;
F6EC      1745 ; zap: nothing
F6EC      1746 ;
F6EC      1747
F6EC CS    1748 SERSEL PUSH BC
F6ED 01 01 OD 1749      LD      BC,13*256+1      ; 300
F6F0 E6 01    1750      AND      1
F6F2 28 03    1751      JR      Z,SERS1
F6F4 01 02 03 1752      LD      BC,3*256+2      ; 1200
F6F7 79      1753 SERS1 LD      A,C
F6F8 32 73 FE 1754      LD      (CSRFF),A
F6FB 78      1755      LD      A,B
F6FC 32 75 FE 1756      LD      (SIC),A
F6FF C1      1757      POP     BC
F700 C9      1758      RET
F701      1759
F701      1760 ;
F701      1761 ; Get input/out routines
F701      1762 ;
F701      1763
F701      1764      COPY SER,IO/I
F701      1765 ;
F701      1766 ; serial port drivers
F701      1767 ;
F701      1768
F701      1769 SERPORT EQU OBEH
F701      1770
F701      1771 ;
F701      1772 ; equate serial status bits for bit, set, res
F701      1773 ;
F701      1774
F701      1775 XSRL EQU 0      ; Xmit Shift Register Loaded
F701      1776 XBRL EQU 1      ; Xmit Buffer Register Loaded
F701      1777 RSRL EQU 2      ; Receive Shift Register Loaded
F701      1778 RBRL EQU 3      ; Receive Buffer Register Loaded
F701      1779 RLSB EQU 4      ; Receive Looking for Start Bit
F701      1780
F701      1781 ;
F701      1782 ; equate serial status bits for masking
F701      1783 ;
F701      1784
F701      1785 MXSRL EQU 001H
F701      1786 MXBRL EQU 002H
F701      1787 MRSRL EQU 004H
F701      1788 MRBRL EQU 008H
F701      1789 MRLSB EQU 010H
F701      1790 ;
F701      1791 ;
F701      1792 ; SERSTAT - return the status of the serial input port
F701      1793 ;
F701      1794 ; usage: call serstat
F701      1795 ;
F701      1796 ; entry: who cares
F701      1797 ;
F701      1798 ; exit: if a character is available res. a contains Offh
F701      1799 ;      else a contains 000h
F701      1800 ;

```

F701		1801	; zaps: a	
F701		1802	;	
F701		1803		
F701	3A 7E FE	1804	SERSTAT LD	A,(SFLQS)
F704	E6 08	1805	AND	MRBRL
F706	C8	1806	RET	Z
F707	3E FF	1807	LD	A,-1
F709	C9	1808	RET	
F70A		1809		
F70A		1810		
F70A		1811	;	
F70A		1812	SERIN - returns a character from the serial port	
F70A		1813	;	
F70A		1814	usage: call serin	
F70A		1815	;	
F70A		1816	entry: who cares	
F70A		1817	;	
F70A		1818	exit: the character is reg a	
F70A		1819	;	
F70A		1820	; zaps: a	
F70A		1821	;	
F70A		1822		
F70A	DD E5	1823	SERIN PUSH	IX
F70C	DD 21 73 FE	1824	LD	IX,SDATA
F710	DD CB 0B 5E	1825	SINI BIT	RBRL,(IX+SFLQS-SDATA)
F714	28 FA	1826	JR	Z,SINI
F716		1827		
F716	3A 78 FE	1828	LD	A,(RBR)
F719	DD CB 0B 9E	1829	RES	RBRL,(IX+SFLQS-SDATA)
F71D	DD E1	1830	POP	IX
F71F	C9	1831	RET	
F720		1832	;	
F720		1833	;	
F720		1834	SEROUT - output the character in c to the serial port	
F720		1835	;	
F720		1836	usage: call serout	
F720		1837	;	
F720		1838	entry: the character is in reg c	
F720		1839	;	
F720		1840	exit: the character is going out the serial	
F720		1841	port with 1 start bit and 1 stop bit	
F720		1842	;	
F720		1843	; zaps Just about everything	
F720		1844	;	
F720		1845		
F720	DB BD	1846	SEROUT IN	A,(RTC)
F722	CB 7F	1847	BIT	7,A
F724	28 FA	1848	JR	Z,SEROUT
F726	3A 7E FE	1849	LD	A,(SFLQS)
F729	CB 4F	1850	BIT	XBRL,A
F72B	20 F3	1851	JR	NZ,SEROUT
F72D	F3	1852	DI	
F72E	CB CF	1853	SET	XBRL,A
F730	32 7E FE	1854	LD	(SFLQS),A
F733	79	1855	LD	A,C
F734	32 7B FE	1856	LD	(XBR),A
F737		1857		
F737	3A 7E FE	1858	LD	A,(SFLQS)
F73A	E6 15	1859	AND	MXSRL+MRLSB+MRSRL ; check for serial activity
F73C	20 12	1860	JR	NZ,S01 ; Jump if any going on

F73E		1861	
F73E E5		1862	PUSH HL
F73F 2A F2 FF		1863	LD HL, (INTV1) ; else save current level 1 int vector
F742 22 BC FE		1864	LD (OIVI), HL
F745 21 38 F0		1865	LD HL, STIMER-P1BIAS ; ...set vector to point to serial timer
F748 22 F2 FF		1866	LD (INTV1), HL
F74B E1		1867	POP HL
F74C		1868	
F74C 3E FF		1869	LD A, -1 ; start rtc to transmit char
F74E D3 BD		1870	OUT (RTC), A
F750		1871	
F750 79	1872 S01	1872	LD A, C
F751 FB		1873	EI
F752 C9		1874	RET
F753		1875	
F753		1876	COPY KBD. IO/1
F753		1877	
F753		1878	; Keyboard io handlers
F753		1879	
F753		1880	
F753		1881	KDATA EQU OBCH
F753		1882	RTC EQU OBDH
F753		1883	
F753		1884	
F753		1885	; KSTAT - return the status of the keyboard
F753		1886	
F753		1887	on entry:
F753		1888	who cares
F753		1889	
F753		1890	on exit:
F753		1891	if there is a character ready register a contains 0ffh
F753		1892	else register a contains 000h
F753		1893	
F753		1894	trashes a
F753		1895	
F753 3A 39 FE		1896	KSTAT LD A, (KDAV)
F756 B7		1897	OR A
F757 C9		1898	RET
F758		1899	
F758		1900	
F758		1901	
F758		1902	; KBREAD - read a character from the keyboard
F758		1903	
F758		1904	on entry:
F758		1905	who cares
F758		1906	
F758		1907	on exit:
F758		1908	the character is in the accumulator
F758		1909	
F758		1910	trashes a
F758		1911	
F758		1912	
F758 CD 53 F7		1913	KBREAD CALL KSTAT
F75B 28 FB		1914	JR Z, KBREAD
F75D AF		1915	XOR A
F75E 32 39 FE		1916	LD (KDAV), A
F761 3A 38 FE		1917	LD A, (KBVAL)
F764 C9		1918	RET
F765		1919	
F765		1920	COPY .PAR. IO/1

```

F765      1921 ;
F765      1922 ; parallel port io handlers
F765      1923 ;
F765      1924
F765      1925 PARDATA EQU 0BFH      ; the data port
F765      1926 PARSTAT EQU 0BEH     ; the status port
F765      1927
F765      1928 PARBUSY EQU 020H     ; parallel device busy
F765      1929 PARIRDY EQU 040H     ; parallel input data ready
F765      1930
F765      1931 ;
F765      1932 ; PSTAT - If input data is available at parallel port returns
F765      1933 ;   Offh in accumulator, else acc. is 0
F765      1934 ;
F765      1935 ;   on entry:
F765      1936 ;   who cares
F765      1937 ;
F765      1938 ;   on exit:
F765      1939 ;   a has ffh & Z flag set if data is ready, else a has 00
F765      1940 ;
F765      1941 ;   trashes a
F765      1942 ;
F765      1943
F765 DB BE      1944 PSTAT IN      A,(PARSTAT)
F767 E6 40      1945      AND      PARIRDY
F769 3E FF      1946      LD      A,OFFH
F76B C8          1947      RET      Z
F76C 2F          1948      CPL
F76D C9          1949      RET
F76E            1950
F76E            1951 ;
F76E            1952 ; PREAD - read a character from the parallel, char returned
F76E            1953 ;   in acc
F76E            1954 ;
F76E            1955 ;   on entry:
F76E            1956 ;   you want a character
F76E            1957 ;
F76E            1958 ;   on exit:
F76E            1959 ;   it's in the accumulator
F76E            1960 ;
F76E            1961 ;   trashes a
F76E            1962 ;
F76E            1963
F76E CD 65 F7   1964 PREAD CALL  PSTAT
F771 B7          1965      OR      A
F772 28 FA      1966      JR      Z,PREAD
F774 DB BF      1967      IN      A,(PARDATA)
F776 C9          1968      RET
F777            1969
F777            1970 ;
F777            1971 ; POUT - output the character in register c to the parallel
F777            1972 ;   port.
F777            1973 ;
F777            1974 ;   on entry:
F777            1975 ;   the character to be output is reg c
F777            1976 ;
F777            1977 ;   on exit:
F777            1978 ;   it's been output
F777            1979 ;
F777            1980 ;   trashes a

```



```

F777      1981 ;
F777      1982 ;
F777 DB BE      1983 POUT IN A, (PARSTAT)
F779 E6 20      1984 AND PARBUSY
F77B 20 FA      1985 JR NZ, POUT
F77D 79          1986 LD A, C
F77E D3 BF      1987 OUT (PARDATA), A
F780 C9          1988 RET
F781          1989 ;
F781          1990 COPY CAS.IO/1
F781          1991 ;
F781          1992 ; equate the cassette io port
F781          1993 ;
F781          1994 ;
F781          1995 CASPORT EQU OBEH
F781          1996 ;
F781          1997 ;
F781          1998 ; equate cassette status bits for bit, set, res
F781          1999 ;
F781          2000 ;
F781          2001 CXSRL EQU 0 ; Cassette Xmit Shift Register Loaded
F781          2002 CXBRL EQU 1 ; Cassette Xmit Buffer Register Loaded
F781          2003 CRSRL EQU 2 ; Cassette Receive Shift Register Loading
F781          2004 CRBRL EQU 3 ; Cassette Receive Buffer Register Loaded
F781          2005 CDATB EQU 4 ; Cassette DATA Bit (type of bit DATA or !CLOCK)
F781          2006 CTOUT EQU 5 ; Cassette TimeOUT flag
F781          2007 CSAWB EQU 7 ; Cassette SAW Bit
F781          2008 ;
F781          2009 ;
F781          2010 ; equate cassette status bits for masking
F781          2011 ;
F781          2012 ;
F781          2013 MCXSRL EQU 001H
F781          2014 MCXBRL EQU 002H
F781          2015 MCRSRL EQU 004H
F781          2016 MCRBRL EQU 008H
F781          2017 MCDATB EQU 010H
F781          2018 MCTOUT EQU 020H
F781          2019 MCRIP EQU 040H
F781          2020 MCSAWB EQU 080H
F781          2021 ;
F781          2022 ;
F781          2023 ; equate masks for cassette states
F781          2024 ;
F781          2025 ;
F781          2026 XCSTATE EQU 0CFH ; mask for removing cassette state
F781          2027 CSTATE EQU 030H ; mask for obtaining cassette state
F781          2028 CLOW EQU 020H ; sets cassette line low
F781          2029 CMID EQU 000H ; sets cassette line to the mid point
F781          2030 CHI EQU 010H ; sets cassette line hi
F781          2031 ;
F781          2032 ;
F781          2033 CASSTAT - return the status of the cassette input port
F781          2034 ;
F781          2035 ; usage: call casstat
F781          2036 ;
F781          2037 ; entry: who cares
F781          2038 ;
F781          2039 ; exit: if a character is available res. a contains Offh
F781          2040 ; else a contains 000h

```

F781	2041 ;		
F781	2042 ; zap5: a		
F781	2043 ;		
F781	2044		
F781 3A 72 FE	2045 CASSTAT LD A,(CFLGS)		
F784 E6 08	2046 AND MCRBRL		
F786 C8	2047 RET Z		
F787 3E FF	2048 LD A,-1		
F789 C9	2049 RET		
F78A	2050		
F78A	2051 ;		
F78A	2052 ; CASIN - returns a character from the casstte port		
F78A	2053 ;		
F78A	2054 ; usage: call casin		
F78A	2055 ;		
F78A	2056 ; entry: who cares		
F78A	2057 ;		
F78A	2058 ; exit: the character is res a		
F78A	2059 ;		
F78A	2060 ; zap5: a		
F78A	2061 ;		
F78A	2062		
F78A DD E5	2063 CASIN PUSH IX		
F78C DD 21 3C FE	2064 LD IX,CDATA		
F790 DD CB 36 5E	2065 CIN1 BIT CRBRL,(IX+CFLOS-CDATA)		
F794 28 FA	2066 JR Z,CIN1		
F796	2067		
F796 3A 70 FE	2068 LD A,(CRBR)		
F799 DD CB 36 9E	2069 RES CRBRL,(IX+CFLOS-CDATA)		
F79D DD E1	2070 POP IX		
F79F C9	2071 RET		
F7A0	2072 ;		
F7A0	2073 ;		
F7A0	2074 ; CASOUT - output the character in c to the cassette port		
F7A0	2075 ;		
F7A0	2076 ; usage: call casout		
F7A0	2077 ;		
F7A0	2078 ; entry: the character is in res c		
F7A0	2079 ;		
F7A0	2080 ; exit: the character is going out the casstte port		
F7A0	2081 ;		
F7A0	2082 ; zap5 Just about everything		
F7A0	2083 ;		
F7A0	2084		
F7A0 3A 72 FE	2085 CASOUT LD A,(CFLOS)		i set cassette flags
F7A3 CB 4F	2086 BIT CXBR,A		i wait for xmit buffer res empty
F7A5 20 F9	2087 JR NZ,CASOUT		
F7A7 F3	2088 DI		
F7A8 CB CF	2089 SET CXBR,A		
F7A8 32 72 FE	2090 LD (CFLOS),A		
F7AD 79	2091 LD A,C		i load Xmit Buffer Register with character
F7AE 32 3E FE	2092 LD (CXBR),A		
F7B1 3A 72 FE	2093 LD A,(CFLGS)		
F7B4 E6 01	2094 AND MCXSR		
F7B6 20 12	2095 JR NZ,C01		
F7B8	2096		
F7B8 3A 7F FE	2097 LD A,(SMASK)		i set cassette state to mid
F7BB E6 CF	2098 AND XCSTATE		
F7BD F6 00	2099 OR CMID		
F7BF 32 7F FE	2100 LD (SMASK),A		

F7C2 AF	2101	XOR	A	; set interval counter to 0
F7C3 32 3C FE	2102	LD	(CXBC),A	
F7C6 3E FF	2103	LD	A,-1	; start the real time clock
F7C8 D3 BD	2104	OUT	(RTC),A	
F7CA	2105			
F7CA 79	2106 C01	LD	A,C	; return the byte in a
F7CB FB	2107	EI		
F7CC C9	2108	RET		
F7CD	2109 ;			
F7CD	2110			
F7CD	2111	ORG	P1ORG-45	
F7D3	2112			
F7D3	2113 ;			
F7D3	2114 ;	POTP1 - Start of page zero to page 1 linkage routine		
F7D3	2115 ;	the rest of the routine is in page 1		
F7D3	2116 ;			
F7D3	2117 ;	usage: call p0tp1		
F7D3	2118 ;			
F7D3	2119 ;	entry: hl has address of the routine you want		
F7D3	2120 ;			
F7D3	2121 ;	zaps: nothing		
F7D3	2122 ;			
F7D3	2123			
F7D3	2124 ;	the following six lines of code are executed in page 0		
F7D3	2125			
F7D3 F3	2126 POTP1 DI			
F7D4 F5	2127	PUSH	AF	; save a and flags
F7D5 3A F0 FE	2128	LD	A,(AUXMASK)	; set current vaule for auxport
F7D8 CB CF	2129	SET	1,A	; switch to bank 1
F7DA 32 F0 FE	2130	LD	(AUXMASK),A	; save the value
F7DD D3 BC	2131	OUT	(KDATA),A	; do the switch
F7DF	2132			
F7DF	2133 ;	fall thru into the code in page 1 (see RPOTP1)		
F7DF	2134			
F7DF	2135 ;			
F7DF	2136 ;	The rest of the code for P1TP0 - executed in page 0		
F7DF	2137 ;			
F7DF	2138			
F7DF F1	2139 RP1TP0 POP	AF		; retrieve acc
F7E0 FB	2140	EI		
F7E1 CD F0 F7	2141	CALL	IYJUMP	; go to the routine in IY
F7E4 F3	2142	DI		
F7E5 F5	2143	PUSH	AF	; save result
F7E6 3A F0 FE	2144	LD	A,(AUXMASK)	; switch back to page 1
F7E9 CB CF	2145	SET	1,A	
F7EB 32 F0 FE	2146	LD	(AUXMASK),A	
F7EE 18 0B	2147	JR	IRE	
F7F0	2148			
F7F0 FD E9	2149 IYJUMP JP	(IY)		
F7F2	2150			
F7F2	2151 ;			
F7F2	2152 ;	interrupt linkage routine		
F7F2	2153 ;			
F7F2	2154			
F7F2 3A F0 FE	2155 IRET0 LD	A,(AUXMASK)		; set current value of aux port
F7F5 CB C7	2156	SET	0,A	; toggle int_svc_done
F7F7 D3 9C	2157	OUT	(KDATA),A	
F7F9 CB 87	2158	RES	0,A	
F7FB D3 BC	2159 IRE	OUT	(KDATA),A	
F7FD F1	2160	POP	AF	

F7FE	FB	2161	EI	
F7FF	C9	2162	RET	
F800		2163		
F800		2164	ORG	P10R0
F800		2165		
F800		2166		
F800		2167		Get the interrupt service routines
F800		2168		
F800		2169		
F800		2170	COPY	RSET.INT/1
F800		2171		
F800		2172		Reset and non-maskable interrupt handlers
F800		2173		
F800		2174		
F800	C3 06 F0	2175	JP	INIT-P1BIAS ; Monitor Cold Start
F803	C3 F2 F7	2176	JP	IRET1-P1BIAS ; Re-entry for user interrupt routines
F806		2177		
F806		2178		initialize the interrupt vectors
F806		2179		
F806	F3	2180	INIT	DI ; actual Monitor Cold Start code starts here
F807	31 C0 FF	2181	LD	SP,MSTACK
F80A	21 78 F0	2182	LD	HL,INTTAB-P1BIAS ; point to initial int vectors
F80D	11 F0 FF	2183	LD	DE,INTVO ; point to start of the interrupt vectors
F810	7A	2184	LD	A,D ; select mode two interrupts
F811	ED 47	2185	LD	I,A
F813	ED 5E	2186	IM	2
F815	01 10 00	2187	LD	BC,8*2 ; copy 8 words
F818	ED B0	2188	LDIR	
F81A		2189		
F81A	21 00 F8	2190	LD	HL,PAGE1 ; init screen cursor address
F81D	22 30 FE	2191	LD	(CURPOS),HL ; cursor address
F820		2192		
F820	3E 80	2193	LD	A,080H
F822	32 F0 FE	2194	LD	(AUXMASK),A
F825		2195		
F825	AF	2196	XOR	A
F826	32 33 FE	2197	LD	(ESCFLA0),A
F829	32 34 FE	2198	LD	(CVAL),A
F82C	32 39 FE	2199	LD	(KDAV),A
F82F	32 37 FE	2200	LD	(VFLOS),A
F832	32 7E FE	2201	LD	(SFLOS),A
F835	32 72 FE	2202	LD	(CFLOS),A
F838	32 3B FE	2203	LD	(RTIME),A
F83B		2204		
F83B	3C	2205	INC	A ; set constant serial fudge factor
F83C	32 73 FE	2206	LD	(CSRFF),A
F83F		2207		
F83F	3E 0D	2208	LD	A,13 ; set serial interval counter to 13 (300 baud)
F841	32 75 FE	2209	LD	(SIC),A
F844		2210		
F844	3E F0	2211	ILOOP	LD A,0F0H ; wait for hardware reset to complete
F846	D3 BD	2212	OUT	(RTC),A
F848	DB BD	2213	IN	A,(RTC)
F84A	E6 3C	2214	AND	03CH ; 3CH allows RTC counter to count 4
F84C	28 F6	2215	JR	Z,ILOOP
F84E	AF	2216	XOR	A
F84F	D3 BD	2217	OUT	(RTC),A
F851		2218		
F851	3E A0	2219	LD	A,0A0H ; enable the serial int. and serial ctrl
F853	D3 BE	2220	OUT	(0BEH),A

F855	32 7F FE	2221	LD	(SMASK),A	; status port stuff
F858		2222			
F859	21 45 F0	2223	LD	HL,START	; return point for reset
F85B	E5	2224	PUSH	HL	
F85C	F5	2225	PUSH	AF	
F85D	C3 F2 F7	2226	JP	IRET1-P1BIAS	
F860		2227			
F860		2228			
F860		2229			; Invalid interrupt handler
F860		2230			
F860		2231			
F860	F5	2232	SPINT PUSH	AF	
F861	C3 F2 F7	2233	JP	IRET1-P1BIAS	
F864		2234			
F864		2235			
F864		2236			; Non-maskable interrupt handler
F864		2237			
F864		2238			
F864		2239	ORG	P10R0+00066H	
F866		2240			
F866	C3 06 F0	2241	JP	INIT-P1BIAS	
F869		2242			
F869		2243			
F869		2244			; Invalid interrupt handler for edge detector
F869		2245			
F869		2246			
F869	F5	2247	SPEINT PUSH	AF	
F86A	3A 7F FE	2248	LD	A,(SMASK)	
F86D	CB BF	2249	RES	7,A	
F86F	D3 BE	2250	OUT	(SERPORT),A	
F871	CB FF	2251	SET	7,A	
F873	D3 BE	2252	OUT	(SERPORT),A	
F875	C3 F2 F7	2253	JP	IRET1-P1BIAS	
F878		2254			
F878		2255			
F878		2256			; INTTAB - The initial interrupt vectors
F878		2257			
F878		2258			
F878	60 F0	2259	INTTAB DW	SPINT-P1BIAS	; interrupt 0 vector
F87A	60 F2	2260	DW	KTIMER-P1BIAS	; RTC interrupt (keyboard repeat timer)
F87C	36 F1	2261	DW	SEINT-P1BIAS	; serial edge detector interrupt
F87E	60 F0	2262	DW	SPINT-P1BIAS	
F880	60 F0	2263	DW	SPINT-P1BIAS	
F882	60 F0	2264	DW	SPINT-P1BIAS	
F884	C1 F1	2265	DW	KBINT-P1BIAS	; keyboard "key pressed" interrupt
F886	60 F0	2266	DW	SPINT-P1BIAS	; interrupt 7 vector
F888		2267			
F888		2268	COPY	SER.INT/1	
F888		2269			
F888		2270			; serial interrupt handlers
F888		2271			
F888		2272			
F888		2273			
F888		2274			; STIMER - serial rtc interrupt handler
F888		2275			
F888		2276			
F888	F5	2277	STIMER PUSH	AF	
F899	DD 22 BA FE	2278	LD	(ISAV1),IX	
F89D		2279			
F89D	DD 21 73 FE	2280	LD	IX,SDATA	

F891		2281	
F891 3A 73 FE		2282	LD A,(CSRFF)
F894 32 74 FE		2283	LD (SRFF),A
F897		2284	
F897 DD CB 0B 46		2285	BIT XSRL,(IX+SFLGS-SDATA) ; check if shift register loaded
F89B 28 48		2286	JR Z,ST10 ; Jump if so
F89D		2287	
F89D		2288	; set here when shift register loaded
F89D		2289	
F89D DD 35 03		2290	DEC (IX+XIC-SDATA) ; decerement xmit interval counter
F8A0 20 5D		2291	JR NZ,ST20 ; Jump if not time to check
F8A2		2292	
F8A2		2293	; set here when time for next bit
F8A2		2294	
F8A2 DD 35 01		2295	ST00 DEC (IX+SRFF-SDATA)
F8A5 20 FB		2296	JR NZ,ST00
F8A7 DD 34 01		2297	INC (IX+SRFF-SDATA)
F8AA		2298	
F8AA DD 35 0A		2299	DEC (IX+XBC-SDATA) ; decrement xmit bit count
F8AD F2 BC F0		2300	JP P,ST01-P1BIAS
F8B0		2301	
F8B0		2302	; set here when done transmitting
F8B0		2303	
F8B0 DD CB 0B 4E		2304	BIT XBRL,(IX+SFLGS-SDATA) ; check if anything in xmit buffer
F8B4 20 35		2305	JR NZ,ST11 ; so start transimtion if XBR not empty
F8B6		2306	
F8B6 DD CB 0B 86		2307	RES XSRL,(IX+SFLGS-SDATA) ; else say shift register empty
F8BA 18 43		2308	JR ST20 ; and so check for input
F8BC		2309	
F8BC 20 0A		2310	ST01 JR NZ,ST02 ; Jump if more data bits
F8BE DD CB 0C A6		2311	RES 4,(IX+SMASK-SDATA) ; else send two stop bits
F8C2 3A 75 FE		2312	LD A,(SIC)
F8C5 87		2313	ADD A,A
F8C6 18 13		2314	JR ST05
F8C8		2315	
F8C8 DD CB 09 0E		2316	ST02 RRC (IX+XSR-SDATA) ; get next bit
F8CC 30 06		2317	JR NC,ST03 ; Jump if bit is low
F8CE DD CB 0C A6		2318	RES 4,(IX+SMASK-SDATA) ; else set serial line hi
F8D2 18 04		2319	JR ST04
F8D4		2320	
F8D4 DD CB 0C E6		2321	ST03 SET 4,(IX+SMASK-SDATA) ; set serial line low
F8D8		2322	
F8D8 3A 75 FE		2323	ST04 LD A,(SIC) ; get serial interval counter
F8DB 32 76 FE		2324	ST05 LD (XIC),A ; set transmit counter
F8DE 3A 7F FE		2325	LD A,(SMASK) ; get value to out to port
F8E1 D3 BE		2326	OUT (SERPORT),A ; out it
F8E3 18 1A		2327	JR ST20
F8E5		2328	
F8E5		2329	; set here to check for char in xmit buffer res
F8E5		2330	
F8E5 DD CB 0B 4E		2331	ST10 BIT XBRL,(IX+SFLGS-SDATA) ; anything in xmit buffer?
F8E9 28 14		2332	JR Z,ST20 ; Jump if not
F8EB		2333	
F8EB DD CB 0B 8E		2334	ST11 RES XBRL,(IX+SFLGS-SDATA) ; say xmit buffer empty
F8EF DD CB 0B C6		2335	SET XSRL,(IX+SFLGS-SDATA) ; say shift register full
F8F3		2336	
F8F3 3A 7B FE		2337	LD A,(XBR) ; transfer xmit buffer to xmit shift register
F8F6 32 7C FE		2338	LD (XSR),A
F8F9		2339	
F8F9 DD 36 0A 09		2340	LD (IX+XBC-SDATA),9 ; set xmit bit count

F8FD 18 D5	2341	JR	ST03	; set interval counter and send start bit
F8FF	2342			
F8FF	2343	; get here to check for input stuff		
F8FF	2344			
F8FF DD CB 0B 66	2345	ST20	BIT	RLSB, (IX+SFLGS-SDATA) ; looking for start bit?
F903 28 25	2346	JR	Z, ST30	; Jump if not
F905	2347			
F905	2348	; get here if looking for start bit		
F905	2349			
F905 DD 35 04	2350	DEC	(IX+RIC-SDATA)	
F908 20 5C	2351	JR	NZ, ST40	; Jump if not time to look for start bit
F90A	2352			
F90A	2353	; set here if time to sample start bit		
F90A	2354			
F90A DB BE	2355	IN	A, (SERPORT)	; sample serial input data
F90C E6 80	2356	AND	080H	; leave inverted data bit
F90E 28 14	2357	JR	Z, ST22	; Jump if no start bit
F910	2358			
F910	2359	; set here when have valid start bit		
F910	2360			
F910 DD CB 0B A6	2361	RES	RLSB, (IX+SFLGS-SDATA)	; looking for start bit = false
F914 DD CB 0B D6	2362	SET	RSRL, (IX+SFLGS-SDATA)	; rec. shift reg. loading = true
F918	2363			
F918 3A 75 FE	2364	LD	A, (SIC)	; set serial interval count
F91B 32 77 FE	2365	LD	(RIC), A	; set receive interval count
F91E	2366			
F91E DD 36 07 09	2367	LD	(IX+RBC-SDATA), 9	; set receive bit count
F922 18 42	2368	JR	ST40	
F924	2369			
F924 DD CB 0B A6	2370	ST22	RES	RLSB, (IX+SFLGS-SDATA) ; wasn't a valid start bit
F928 18 33	2371	JR	ST33	
F92A	2372			
F92A	2373	; set here if not looking for start bit		
F92A	2374			
F92A DD CB 0B 56	2375	ST30	BIT	RSRL, (IX+SFLGS-SDATA) ; rec. shift register loading?
F92E 28 36	2376	JR	Z, ST40	; Jump if not
F930	2377			
F930	2378	; set here if loading receive shift register		
F930	2379			
F930 DD 35 04	2380	DEC	(IX+RIC-SDATA)	; decrement receive interval counter
F933 20 31	2381	JR	NZ, ST40	
F935	2382			
F935	2383	; set here if time to sample data bit		
F935	2384			
F935 DD 35 01	2385	ST31	DEC	(IX+SRFF-SDATA)
F938 20 FB	2386	JR	NZ, ST31	
F93A	2387			
F93A DD 35 07	2388	DEC	(IX+RBC-SDATA)	; decrement receive bit count
F93D 28 10	2389	JR	Z, ST32	; Jump if received all bits
F93F	2390			
F93F DB BE	2391	IN	A, (SERPORT)	; sample data bit
F941 07	2392	RLCA		; rotate bit to carry
F942 3F	2393	CCF		; the data bit is inverted, so complement carry
F943 DD CB 06 1E	2394	RR	(IX+RSR-SDATA)	; rotate bit into shift register
F947	2395			
F947 3A 75 FE	2396	LD	A, (SIC)	; set serial interval counter
F94A 32 77 FE	2397	LD	(RIC), A	; ... to receive interval counter
F94D 18 17	2398	JR	ST40	
F94F	2399			
F94F	2400	; set here when received all bits		

F94F		2401			
F94F 3A 79 FE	2402	ST32	LD	A, (RSR)	; set received data...
F952 32 78 FE	2403		LD	(RBR), A	; to receive buffer register
F955	2404				
F955 DD CB 0B DE	2405		SET	RBRL, (IX+SFLGS-SDATA)	; set receive buffer register loaded
F959 DD CB 0B 96	2406		RES	RSRL, (IX+SFLGS-SDATA)	; RSRL = false
F95D	2407				
F95D DD CB 0C FE	2408	ST33	SET	7, (IX+SMASK-SDATA)	; reenable the serial interrupt
F961 3A 7F FE	2409		LD	A, (SMASK)	
F964 D3 BE	2410		OUT	(SERPORT), A	
F966	2411				
F966 3A 7E FE	2412	ST40	LD	A, (SFLGS)	; check if we should reenable the clock
F969 E6 15	2413		AND	MXSRL+MRLSB+MRSRL	
F96B 20 0E	2414		JR	NZ, ST50	; Jump if not done with all serial stuff
F96D	2415				
F96D DD 2A BC FE	2416		LD	IX, (OIVI)	; restore old level 1 int vector
F971 DD 22 F2 FF	2417		LD	(INTV1), IX	
F975	2418				
F975 3A 3B FE	2419		LD	A, (RTIME)	; check if we should restart the clock
F978 B7	2420		OR	A	; for the keyboard repeat function
F979 28 02	2421		JR	Z, ST60	; if counter is zero then don't restart
F97B	2422				
F97B 3E FD	2423	ST50	LD	A, -3	restart the real time clock
F97D D3 BD	2424	ST60	OUT	(RTC), A	; stop real time clock
F97F DD 2A BA FE	2425		LD	IX, (ISAV1)	
F983 C3 F2 F7	2426		JP	IRET1-P1BIAS	
F986	2427				
F986	2428				
F986	2429				
F986	2430				
F986	2431				
F986 F5	2432	SEINT	PUSH	AF	; save a and flags
F987	2433				
F987 3A 7F FE	2434		LD	A, (SMASK)	; disable edge detector interrupt
F98A CB BF	2435		RES	7, A	
F98C 32 7F FE	2436		LD	(SMASK), A	
F98F D3 BE	2437		OUT	(SERPORT), A	
F991	2438				
F991 3A 75 FE	2439		LD	A, (SIC)	; set serial interval counter
F994 CB 3F	2440		SRL	A	; divide by two for half bit time
F996 32 77 FE	2441		LD	(RIC), A	; set receive interval counter
F999	2442				
F999 3A 7E FE	2443		LD	A, (SFLGS)	
F99C E6 15	2444		AND	MXSRL+MRLSB+MRSRL	; check for serial activity
F99E 20 16	2445		JR	NZ, SE1	; Jump if any going on
F9A9	2446				
F9A9 22 BA FE	2447		LD	(ISAV1), HL	
F9A3 2A F2 FF	2448		LD	HL, (INTV1)	; else save current level 1 int vector
F9A6 22 80 FE	2449		LD	(OIVI), HL	
F9A9 21 88 F0	2450		LD	HL, STIMER-P1BIAS	; ...set vector to point to serial timer
F9AC 22 F2 FF	2451		LD	(INTV1), HL	
F9AF 2A BA FE	2452		LD	HL, (ISAV1)	
F9B2	2453				
F9B2 3E FD	2454		LD	A, -3	; set clock value
F9B4 D3 BD	2455		OUT	(RTC), A	
F9B6	2456				
F9B6 3A 7E FE	2457	SE1	LD	A, (SFLGS)	
F9B9 CB E7	2458		SET	RLSB, A	
F9BB 32 7E FE	2459		LD	(SFLGS), A	
F9BE C3 F2 F7	2460		JP	IRET1-P1BIAS	

F9C1	2461 ;			
F9C1	2462	COPY	KBD.INT/1	
F9C1	2463 ;			
F9C1	2464 ;	keyboard interrupt service routines		
F9C1	2465 ;			
F9C1	2466			
F9C1	2467 ;	define repeat rate for new key down		
F9C1	2468			
F9C1	2469	KRD1 EQU	187	; initial repeat time 187 * 4 = 748 msec
F9C1	2470	KRD2 EQU	13	; secondary repeat value 13 * 4 = 50 msec
F9C1	2471			
F9C1	2472 ;	define the keyboard hardware		
F9C1	2473			
F9C1	2474	NPBIT EQU	2	; if this bit is high then key is on numeric pad
F9C1	2475			
F9C1	2476	KBSTAT EQU	OBEH	
F9C1	2477			
F9C1	2478	SKDOWN EQU	0	; low when shift key down
F9C1	2479	CKDOWN EQU	1	; low when control key down
F9C1	2480	UKDOWN EQU	2	; low when upper case key down
F9C1	2481	SWEDSH EQU	3	; low if swedish style keyboard
F9C1	2482 ;			
F9C1	2483 ;			
F9C1	2484 ;	KBINT - the keyboard interrupt handler. Process decoding		
F9C1	2485 ;	of the keyboard when a key is hit		
F9C1	2486 ;			
F9C1	2487 ;	on entry:		
F9C1	2488 ;	someone poked a key		
F9C1	2489 ;			
F9C1	2490 ;	on exit:		
F9C1	2491 ;	the decoded value of the key is stored in kbval		
F9C1	2492 ;			
F9C1	2493 ;	trashes nothing		
F9C1	2494 ;			
F9C1	2495			
F9C1 F5	2496	KBINT PUSH	AF	; save acc and flags
F9C2 22 BA FE	2497	LD	(ISAV1),HL	
F9C5	2498			
F9C5 DB BC	2499	IN	A,(KDATA)	
F9C7 E6 7F	2500	AND	07FH	
F9C9 32 3A FE	2501	LD	(KADR),A	; save a copy of the address
F9CC	2502			
F9CC CB 57	2503	BIT	NPBIT,A	; check if key is on the numeric pad
F9CE 28 10	2504	JR	Z,NOTNP	
F9D0	2505			
F9D0 0F	2506	RRCA		; rotate column to correct position
F9D1 0F	2507	RRCA		
F9D2 0F	2508	RRCA		
F9D3 E6 0F	2509	AND	00FH	; keep it legal
F9D5 21 89 F2	2510	LD	HL,NPMAP-PIBIAS	; point to numeric pad map
F9D8 85	2511	ADD	A,L	
F9D9 6F	2512	LD	L,A	
F9DA 30 01	2513	JR	NC,KBIO	
F9DC 24	2514	INC	H	
F9DD 7E	2515	KBIO LD	A,(HL)	; set the character
F9DE 18 5B	2516	JR	KB14	
F9E0	2517			
F9E0 E6 03	2518	NOTNP AND	003H	; isolate the row bits
F9E2 67	2519	LD	H,A	
F9E3 3A 3A FE	2520	LD	A,(KADR)	; set new copy of key address

F9E6 0F	2521	RRCA			; shift column bits down one
F9E7 E6 3C	2522	AND	03CH		; leave the column bits
F9E9 B4	2523	OR	H		; or in the row bits
F9EA 67	2524	LD	H,A		; save in H
F9EB DB BE	2525	IN	A,(KBSTAT)		; check for swedish keyboard
F9ED CB 5F	2526	BIT	SWEDSH,A		
F9EF 7C	2527	LD	A,H		; set key address to A
F9F0 21 99 F2	2528	LD	HL,UKMAP-P1BIAS		; index into the US key table
F9F3 20 03	2529	JR	NZ,KBI00		; if bit is hi then it's a US keyboard
F9F5 21 19 F3	2530	LD	HL,SKMAP-P1BIAS		; else index into SW key table
F9F8 87	2531 KBI00	ADD	A,A		; multiply by 2
F9F9 35	2532	ADD	A,L		
F9FA 6F	2533	LD	L,A		
F9FB 30 01	2534	JR	NC,KBI1		
F9FD 24	2535	INC	H		
F9FE	2536				
F9FE DB BE	2537 KBI1	IN	A,(KBSTAT)		; check for shift key down
FA00 CB 47	2538	BIT	SKDOWN,A		
FA02 20 03	2539	JR	NZ,KBI2		
FA04	2540				
FA04 23	2541	INC	HL		; point to next byte in table
FA05 18 1D	2542	JR	KBI3		
FA07	2543				
FA07	2544				
FA07 CB 57	2545 KBI2	BIT	UKDOWN,A		; check for upper case key down
FA09 20 23	2546	JR	NZ,KBI32		; Jump if not
FA0B	2547				
FA0B CB 5F	2548	BIT	SWEDSH,A		; check for swedish keyboard
FA0D 7E	2549	LD	A,(HL)		
FA0E 20 14	2550	JR	NZ,KBI3		; Jump if not
FA10	2551				
FA10 FE 7C	2552	CP	'I'		; check for special swedish characters
FA12 28 19	2553	JR	Z,KBI31		
FA14 FE 7D	2554	CP	'J'		
FA16 28 15	2555	JR	Z,KBI31		
FA18 FE 7B	2556	CP	'['		
FA1A 28 11	2557	JR	Z,KBI31		
FA1C FE 60	2558	CP	'\"'		
FA1E 28 0D	2559	JR	Z,KBI31		
FA20 FE 7E	2560	CP	'\"'		
FA22 28 09	2561	JR	Z,KBI31		
FA24	2562				
FA24 7E	2563 KBI3	LD	A,(HL)		; else check for normal upper case'able
FA25 FE 61	2564	CP	'a'		
FA27 38 05	2565	JR	C,KBI32		
FA29 FE 7B	2566	CP	'z'+1		
FA2B 30 01	2567	JR	NC,KBI32		
FA2D	2568				
FA2D 23	2569 KBI31	INC	HL		
FA2E	2570				
FA2E DB BE	2571 KBI32	IN	A,(KBSTAT)		
FA30 CB 4F	2572	BIT	CKDOWN,A		; check for control key down
FA32 7E	2573	LD	A,(HL)		
FA33 20 06	2574	JR	NZ,KBI4		; Jump if not
FA35 FE 20	2575	CP	' '		
FA37 28 02	2576	JR	Z,KBI4		; space can't be controled either
FA39	2577				
FA39 E6 1F	2578	AND	01FH		
FA3B	2579				
FA3B 32 38 FE	2580 KBI4	LD	(KBVAL),A		; save the character


```

FA3E      2581
FA3E 3E FF      2582      LD      A,-1
FA40 32 39 FE    2583      LD      (KDAV),A      ; set the data available flag
FA43      2584
FA43 3E BB      2585      LD      A,KRD1
FA45 32 3B FE    2586      LD      (RTIME),A      ; set the initial time delay
FA48      2587
FA48 3A 7E FE    2588      LD      A,(SFLGS)      ; start the clock if there's not any
FA4B E6 15      2589      AND     MX3RL+MRLSB+MRSRL
FA4D 20 0B      2590      JR      NZ,KBI5      ; jump if any serial stuff
FA4F 3A 72 FE    2591      LD      A,(CFLGS)
FA52 E6 40      2592      AND     MCRIP
FA54 20 04      2593      JR      NZ,KBI5
FA56 3E 80      2594      LD      A,128      ; start the real time clock for maximum time
FA58 D3 BD      2595      OUT     (RTC),A
FA5A 2A BA FE    2596 KBI5 LD      HL,(ISAV1)
FA5D C3 F2 F7    2597      JP      IRET1-P1BIAS
FA60      2598 ;
FA60      2599 ;
FA60      2600 ; KTIMER - process timer interrupt for keyboard repeat funct
FA60      2601 ;
FA60      2602
FA60 F5         2603 KTIMER PUSH AF
FA61 3E 80      2604      LD      A,128
FA63 D3 BD      2605      OUT     (RTC),A      ; restart the timer
FA65 3A 3B FE    2606      LD      A,(RTIME)      ; set timer counter
FA68 3D         2607      DEC     A      ; decrement it
FA69 20 17      2608      JR      NZ,KTM1      ; jump if not done yet
FA6B      2609
FA6B 3A 3A FE    2610      LD      A,(KADR)      ; timer timed out...
FA6E D3 BC      2611      OUT     (KDATA),A      ; check if last key still down
FA70 DB BC      2612      IN      A,(KDATA)
FA72 E6 80      2613      AND     080H      ; mask out hi bit
FA74 20 05      2614      JR      NZ,KTMO      ; jump if key still down
FA76      2615
FA76 AF         2616      XOR     A
FA77 D3 BD      2617      OUT     (RTC),A      ; stop the timer if key not down
FA79 18 07      2618      JR      KTM1
FA7B      2619
FA7B 3E FF      2620 KTMO LD      A,-1
FA7D 32 39 FE    2621      LD      (KDAV),A      ; set keyboard data available flag
FA80 3E 0D      2622      LD      A,KRD2      ; set current secondary delay value
FA82      2623
FA82 32 3B FE    2624 KTM1 LD      (RTIME),A      ; stop the counter if key no longer down
FA85 C3 F2 F7    2625      JP      IRET1-P1BIAS
FA88 FF         2626      DB      OFFH
FA89      2627 ;
FA89      2628 ;
FA89      2629 ; define the numeric pad key values
FA89      2630 ;
FA89      2631
FA89      2632 NPMAP EQU $      ; define the numeric pad keys
FA89      2633
FA89 30 31 32 33 2634      DB      '0','1','2','3'      ; cols 0-3
FA8D 34 35 36 37 2635      DB      '4','5','6','7'      ; cols 4-7
FA91 38 39 2E 3F 2636      DB      '8','9','.','?'      ; cols 8-11
FA95 2B 2D 2A 2F 2637      DB      '+','-','*','/'      ; cols 12-15
FA99      2638 ;
FA99      2639 ; KMAP - keyboard mapping tables
FA99      2640 ;

```

```

FA99      2641 : each entry is associated with one key ( unshifted/shifted )
FA99      2642 ;
FA99      2643 : the table is mapped in the following manner:
FA99      2644 ;
FA99      2645 : the ascii value of a key is derived by adding 2 times the
FA99      2646 : row/column address read from the keyboard to the base
FA99      2647 : address of kmap. if an entry in the table is encoded
FA99      2648 : with the hi bit set then that entry must receive special
FA99      2649 : attention when shifting or controlling the key
FA99      2650
FA99      2651 : define the fill value for the map
FA99      2652 : used to mark invalid keys
FA99      2653
FA99      2654 FILL EQU OFFH
FA99      2655
FA99      2656 : define the special key flag
FA99      2657
FA99      2658 SPECIAL EQU 080H
FA99      2659
FA99      2660 : define the program keys
FA99      2661
FA99      2662 P001 EQU 000H+SPECIAL
FA99      2663 P023 EQU 002H+SPECIAL
FA99      2664
FA99      2665 : define special keys on main keyboard
FA99      2666
FA99      2667 KBTAB EQU 009H
FA99      2668 KBESC EQU 01BH
FA99      2669 RETURN EQU 00DH
FA99      2670 SPACE EQU 020H
FA99      2671 CUNDL EQU 05FH
FA99      2672 ;
FA99      2673 ;
FA99      2674 : this is the map of the USA expander keyboard
FA99      2675 ;
FA99      2676
FA99      2677 : map the keys on the main keyboard
FA99      2678
FA99      2679 UKMAP EQU $
FA99      2680
FA99      2681 DB KBESC,KBESC ; col 0
FA99      2682 DB KBTAB,KBTAB
FA99      2683 DB FILL,FILL
FA99      2684 DB P001,P001+1
FA99      2685
FA99      2686 ASC '1!qQaAzZ' ; col 1
FA99      2687
FA99      2688 ASC '2"WwSsXx' ; col 2
FA99      2689
FA99      2690 ASC '3#eEdDcC' ; col 3
FA99      2691
FA99      2692 ASC '4*rRfFvV' ; col 4
FA99      2693
FA99      2694 ASC '5%tTsObB' ; col 5
FA99      2695

```

FAC9 36 26 79 59	2696	ASC	'6&yYhHnN'	1 col 6
68 48 6E 4E				
FAD1	2697			
FAD1 37 27 75 55	2698	ASC	"7'uUJJimM"	1 col 7
6A 4A 6D 4D				
FAD9	2699			
FAD9 38 28 69 49	2700	ASC	'8(1IkK,<'	1 col 8
6B 4B 2C 3C				
FAE1	2701			
FAE1 39 29 6F 4F	2702	ASC	'9)00IL.>'	1 col 9
6C 4C 2E 3E				
FAE9	2703			
FAE9 30 20 70 50	2704	ASC	'0	pPi+/?' 1 col 10
3B 2B 2F 3F				
FAF1	2705			
FAF1 2D 3D 40 60	2706	ASC	'--@:!*'	1 col 11
3A 2A				
FAF7 20 20	2707	DB	SPACE,SPACE	
FAF9	2708			
FAF9 5E 7E	2709	ASC	'^~/'	1 col 12
FAFB 0D 0D	2710	DB	RETURN,RETURN	
FAFD 5F 7F	2711	DB	CUNDL,CDEL	
FAFF FF FF	2712	DB	FILL,FILL	
FB01	2713			
FB01 5B 7B	2714	ASC	'[[['	1 col 13
FB03 0D 0D	2715	DB	RETURN,RETURN	
FB05 0B 0B	2716	DB	CUP,CUP	
FB07 08 08	2717	DB	CLEFT,CLEFT	
FB09	2718			
FB09 5C 7C	2719	ASC	'\I'	1 col 14
FB0B 82 83	2720	DB	PG23,PG23+1	
FB0D 0A 0A	2721	DB	CLF,CLF	
FB0F 0C 0C	2722	DB	CRIGHT,CRIGHT	
FB11	2723			
FB11 5D 7D	2724	ASC	'J)'	1 col 15
FB13 FF FF	2725	DB	FILL,FILL	
FB15 FF FF	2726	DB	FILL,FILL	
FB17 FF FF	2727	DB	FILL,FILL	
FB19	2728			
FB19	2729			
FB19	2730			
FB19	2731			
FB19	2732			
FB19	2733	SKMAP EQU	\$	
FB19	2734			
FB19 1B 1B	2735	DB	KBESC,KBESC	1 col 0
FB1B 09 09	2736	DB	KBTAB,KBTAB	
FB1D FF FF	2737	DB	FILL,FILL	
FB1F 80 81	2738	DB	PG01,PG01+1	
FB21	2739			
FB21 31 21 71 51	2740	ASC	'1!~QaAzZ'	1 col 1
61 41 7A 5A				
FB29	2741			
FB29 32 22 77 57	2742	ASC	'2"~W~S~X~X'	1 col 2
73 53 78 58				
FB31	2743			
FB31 33 23 65 45	2744	ASC	'3~eEdDcC'	1 col 3
64 44 63 43				
FB39	2745			
FB39 34 24 72 52	2746	ASC	'4~rRfFvV'	1 col 4 # is "circle X"

66 46 76 56			
FB41	2747		
FB41 35 25 74 54	2748	ASC	'5%tTgObB' ; col 5
67 47 62 42			
FB49	2749		
FB49 36 26 79 59	2750	ASC	'6&yYhHnN' ; col 6
68 48 6E 4E			
FB51	2751		
FB51 37 2F 75 55	2752	ASC	"7/uUJUmM" ; col 7
6A 4A 6D 4D			
FB59	2753		
FB59 38 28 69 49	2754	ASC	'8(iIkK,i' ; col 8
6B 4B 2C 3B			
FB61	2755		
FB61 39 29 6F 4F	2756	ASC	'9)oOIL.: ' ; col 9
6C 4C 2E 3A			
FB69	2757		
FB69 30 3D 70 50	2758	ASC	'0=pPI\-' ; col 10 ; is o w/ umlaut, \ is O w/ umlaut
7C 5C 2D			
FB70 5F	2759	DB	CUNDL
FB71	2760		
FB71 2B 3F 7D 5D	2761	ASC	'+?)J([' ; col 11 ; is a with circle, J is A w/ circle
7B 5B			
FB77 20 20	2762	DB	SPACE,SPACE ; ; is a w/ umlaut, [is A w/ umlaut
FB79	2763		
FB79 60 40	2764	ASC	'^@' ; col 12 ; is e accent, @ is E accent
FB7B 0D 0D	2765	DB	RETURN,RETURN
FB7D 27 2A	2766	ASC	"/*"
FB7F FF FF	2767	DB	FILL,FILL
FB81	2768		
FB81 7E 5E	2769	ASC	'^~' ; col 13 ; is u with umlaut, ^ is U w/ umlaut
FB83 0D 0D	2770	DB	RETURN,RETURN
FB85 0B 0B	2771	DB	CUP,CUP
FB87 0B 0B	2772	DB	CLEFT,CLEFT
FB89	2773		
FB89 3E 3C	2774	ASC	'><' ; col 14
FB8B 82 83	2775	DB	P023,P023+1
FB8D 0A 0A	2776	DB	CLF,CLF
FB8F 0C 0C	2777	DB	CRIGHT,CRIGHT
FB91	2778		
FB91 7F 7F	2779	DB	CDEL,CDEL ; col 15
FB93 FF FF	2780	DB	FILL,FILL
FB95 FF FF	2781	DB	FILL,FILL
FB97 FF FF	2782	DB	FILL,FILL
FB99	2783		
FB99	2784	COPY	CAS.INT/1
FB99	2785		
FB99	2786		
FB99	2787		
FB99	2788		
FB99 F5	2789	CXTIMER PUSH AF	; save flags and a
FB9A DD 22 BA FE	2790	LD	(ISAV1),IX
FB9E	2791		
FB9E DD 21 3C FE	2792	LD	IX,CDATA ; load cassette stuff address
FBA2	2793		
FBA2 3E F0	2794	LD	A,-16
FBA4 D3 BD	2795	OUT	(RTC),A
FBA6	2796		
FBA6 DD 35 00	2797	DEC	(IX+CXBC-CDATA)
FBA9 F2 CD F3	2798	JP	P,CXT2-P1BIAS

FBAC		2799			
FBAC DD CB 36 86	2800	RES	CXSRL,(IX+CFLGS-CDATA)	; say shift register not loaded	
FBB0 DD CB 36 4E	2801	BIT	CXBRL,(IX+CFLGS-CDATA)	; anything in the xmit buffer?	
FBB4 20 05	2802	JR	NZ,CXT1		
FBB6	2803				
FBB6 AF	2804	XOR	A		
FBB7 D3 BD	2805	OUT	(RTC),A		
FBB9 18 43	2806	JR	CXTDONE		
FBBB	2807				
FBBB DD CB 36 8E	2808	CXT1 RES	CXBRL,(IX+CFLGS-CDATA)	; say buffer reg not loaded	
FBBF DD CB 36 C6	2809	SET	CXSRL,(IX+CFLGS-CDATA)	; say shift register loaded	
FBC3 3A 3E FE	2810	LD	A,(CXBRL)	; set value in xmit buffer register	
FBC6 32 3F FE	2811	LD	(CXSRL),A	; ...to the shift register	
FBC9 DD 36 00 0F	2812	LD	(IX+CXBC-CDATA),15	; set counter to 15 to force clock	
FBCD	2813				
FBCD DD CB 00 46	2814	CXT2 BIT	0,(IX+CXBC-CDATA)		
FBD1 20 06	2815	JR	NZ,CXT3		
FBD3	2816				
FBD3 DD CB 03 06	2817	RLC	(IX+CXSRL-CDATA)	; then send next data bit, set the bit	
FBD7 30 25	2818	JR	NC,CXTDONE	; Jump if the next bit is zero	
FBD9	2819				
FBD9 3A 7F FE	2820	CXT3 LD	A,(SMASK)	; else set the port value	
FBD0 E6 CF	2821	AND	XCSTATE	; knock out the cassette bits	
FBE0 F6 10	2822	OR	CHI	; set the line hi	
FBE0 D3 BE	2823	OUT	(CASPORT),A		
FBE2	2824				
FBE2 3E 28	2825	LD	A,40	; delay a while	
FBE4 3D	2826	CXT4 DEC	A		
FBE5 20 FD	2827	JR	NZ,CXT4		
FBE7	2828				
FBE7 3A 7F FE	2829	LD	A,(SMASK)		
FBEA E6 CF	2830	AND	XCSTATE	; set the line low	
FBE0 F6 20	2831	OR	CLOW		
FBE0 D3 BE	2832	OUT	(CASPORT),A		
FBE0	2833				
FBE0 3E 28	2834	LD	A,40	; delay a while	
FBE2 3D	2835	CXT5 DEC	A		
FBE3 20 FD	2836	JR	NZ,CXT5		
FBE5	2837				
FBE5 3A 7F FE	2838	LD	A,(SMASK)		
FBE8 E6 CF	2839	AND	XCSTATE	; set the line mid	
FBEA F6 00	2840	OR	CMID		
FBE0 D3 BE	2841	OUT	(CASPORT),A		
FBE0	2842				
FBE0 DD 2A BA FE	2843	CXTDONE LD	IX,(ISAV1)		
FC02 C3 F2 F7	2844	JP	IRET1-PIBIAS		
FC05	2845				
FC05	2846				
FC05	2847				
FC05	2848				
FC05	2849				
FC05 F5	2850	CRTIMER PUSH AF		; save flags	
FC06 DD 22 BA FE	2851	LD	(ISAV1),IX	; and ix reg	
FC0A	2852				
FC0A DD 21 3C FE	2853	LD	IX,CDATA	; point to cassette data storage	
FC0E	2854				
FC0E AF	2855	XOR	A	; stop the real time clock	
FC0F D3 BD	2856	OUT	(RTC),A		
FC11	2857				
FC11 DD CB 36 16	2858	RL	(IX+CFLGS-CDATA)	; shift any data bit to carry	

FC15 DD CB 35 16	2859	RL	(IX+CRSR-CDATA) ; and rotate it into the shift res
FC19 DD CB 36 3E	2860	SRL	(IX+CFLGS-CDATA) ; clear saw data bit
FC1D DD CB 36 A6	2861	RES	CDATB,(IX+CFLGS-CDATA) ; reset data bit flag
FC21 DD 35 01	2862	DEC	(IX+CRBC-CDATA) ; decrement receive bit count
FC24 F2 31 F4	2863	JP	P,CRT1-P1BIAS ; Jump if not done
FC27	2864		
FC27 DD CB 36 D6	2865	SET	CRSRL,(IX+CFLGS-CDATA) ; say shift res loading
FC2B DD 36 01 07	2866	LD	(IX+CRBC-CDATA),7 ; set bit count
FC2F 18 10	2867	JR	CRT2 ; and wait for clock bit
FC31	2868		
FC31 20 0E	2869	CRT1 JR	NZ,CRT2 ; Jump if not last bit
FC33 3A 71 FE	2870	LD	A,(CRSR) ; set contents of shift register
FC36 32 70 FE	2871	LD	(CRBR),A ; ...to buffer register
FC39 DD CB 36 DE	2872	SET	CRBRL,(IX+CFLGS-CDATA) ; say recv. buf. res. loaded
FC3D DD CB 36 96	2873	RES	CRSRL,(IX+CFLGS-CDATA) ; recv. shift res not loading
FC41	2874		
FC41 DD 2A BA FE	2875	CRT2 LD	IX,(ISAV1) ; retrieve ix
FC45 C3 F2 F7	2876	JP	IRET1-P1BIAS
FC48	2877		
FC48	2878		
FC48	2879		EDGE - Cassette edge detector interrupt processor
FC48	2880		
FC48	2881		
FC48 F5	2882	CEDGE PUSH AF	; save flags
FC49 DD 22 BA FE	2883	LD	(ISAV1),IX ; and ix
FC4D	2884		
FC4D DD 21 3C FE	2885	LD	IX,CDATA ; point to cassette data storage
FC51	2886		
FC51 DD CB 36 66	2887	BIT	CDATB,(IX+CFLGS-CDATA) ; check bit type (data or !clock)
FC55 20 0A	2888	JR	NZ,CE1 ; Jump if bit type is data
FC57	2889		
FC57 DD CB 36 E6	2890	SET	CDATB,(IX+CFLGS-CDATA) ; else say next bit is data
FC5B 3E ED	2891	LD	A,-19 ; start realtime clock for ~1.2 msec
FC5D D3 BD	2892	OUT	(RTC),A
FC5F 18 04	2893	JR	CE2 ; and done
FC61	2894		
FC61 DD CB 36 FE	2895	CE1 SET	CSAWB,(IX+CFLGS-CDATA) ; say saw bit
FC65	2896		
FC65 3A 7F FE	2897	CE2 LD	A,(SMASK) ; reenale the edge detector interrupt
FC68 CB BF	2898	RES	7,A
FC6A D3 BE	2899	OUT	(CASPORT),A
FC6C CB FF	2900	SET	7,A
FC6E 32 7F FE	2901	LD	(SMASK),A
FC71 D3 BE	2902	OUT	(CASPORT),A
FC73 DD 2A BA FE	2903	LD	IX,(ISAV1)
FC77 C3 F2 F7	2904	JP	IRET1-P1BIAS
FC7A	2905		
FC7A	2906		
FC7A	2907		
FC7A	2908		CSEDGE - Cassette sync edge detector
FC7A	2909		Only valid while executing SYNC
FC7A	2910		
FC7A	2911		
FC7A F5	2912	CSEDGE PUSH AF	
FC7B	2913		
FC7B 3A 7F FE	2914	LD	A,(SMASK) ; reenale edge detector
FC7E CB BF	2915	RES	7,A
FC80 D3 BE	2916	OUT	(CASPORT),A
FC82 CB FF	2917	SET	7,A
FC84 32 7F FE	2918	LD	(SMASK),A

FC87 D3 BE	2919	OUT	(CASPORT),A	
FC89 CB FB	2920	SET	CSAWB,E	; say we saw a bit
FC8B C3 F2 F7	2921	JP	IRET1-P1BIAS	
FC8E	2922			
FC8E	2923			
FC8E	2924			
FC8E	2925			
FC8E	2926			
FC8E	2927			
FC8E F5	2928	CSTIMER	PUSH AF	
FC9F AF	2929	XOR	A	; stop the realtime clock
FC90 D3 BD	2930	OUT	(RTC),A	
FC92 CB 03	2931	RLC	E	; rotate saw bit flag to carry...
FC94 CB 12	2932	RL	D	; and into byte accumulator
FC96 CB EB	2933	SET	CTOUT,E	; say we timed out
FC98 C3 F2 F7	2934	JP	IRET1-P1BIAS	
FC9B	2935			
FC9B	2936	COPY	P1ENT.CD/1	
FC9B	2937			
FC9B	2938			
FC9B	2939			
FC9B	2940			
FC9B FD 21 ED F2	2941	P1ENTER LD	IY,GETADDR	; set original enter address
FC9F CD D3 F7	2942	CALL	P1TPO-P1BIAS	
FCA2 D8	2943	RET	C	
FCA3 D5	2944	PUSH	DE	; move it to ix
FCA4 DD E1	2945	POP	IX	
FCA6	2946			
FCA6 21 ED F4	2947	ENTR LD	HL,P1CRLF-P1BIAS	
FCA9 CD 5D F6	2948	CALL	P1PUTS-P1BIAS	
FCAC	2949			
FCAC FD 21 C5 F2	2950	LD	IY,GETLINE	; set line of data
FCB0 CD D3 F7	2951	CALL	P1TPO-P1BIAS	
FCB3	2952			
FCB3 2A 30 FE	2953	LD	HL,(CURPOS)	; zap the cursor
FCB6 3A 32 FE	2954	LD	A,(WASTHERE)	
FCB9 77	2955	LD	(HL),A	
FCBA	2956			
FCBA FD 21 B1 F5	2957	LD	IY,ATOLC	; calculate screen address of start of line
FCBE CD D3 F7	2958	CALL	P1TPO-P1BIAS	
FCC1 79	2959	LD	A,C	; check if no data (ie. Just a carriage return)
FCC2 B7	2960	OR	A	
FCC3 C8	2961	RET	Z	; end of enter if so
FCC4	2962			
FCC4 0E 00	2963	LD	C,0	; else set cursor to start of the line
FCC6 FD 21 88 F5	2964	LD	IY,LCTOA	
FCCA CD D3 F7	2965	CALL	P1TPO-P1BIAS	
FCCD	2966			
FCCD FD 21 F3 F2	2967	EN1 LD	IY,OHWORD	; set the next word
FCD1 CD D3 F7	2968	CALL	P1TPO-P1BIAS	
FCD4 30 06	2969	JR	NC,EN2	; Jump if no error
FCD6	2970			
FCD6 FE 20	2971	CP	' '	; check if error was no data word
FCD8 28 CC	2972	JR	Z,ENTR	; set next line if so
FCDA 37	2973	SCF		
FCDB C9	2974	RET		; else error
FCDC	2975			
FCDC FE 3A	2976	EN2 CP	' '	; new address specified?
FCDE 20 06	2977	JR	NZ,EN3	; Jump if so
FCE0 23	2978	INC	HL	; skip the colon

FCE1 06	2979	PUSH DE	; and set new address to ix
FCE2 DD E1	2980	POP IX	
FCE4 18 E7	2981	JR EN1	; set next word
FCE6	2982		
FCE6 DD 73 00	2983	EN3 LD (IX+0),E	; else Plop data into memory
FCE9 DD 23	2984	INC IX	; increment pointer
FCEB 18 E0	2985	JR EN1	
FCED	2986		
FCED 0D 0A 00	2987	PICRLF DB 00DH,00AH,000H	
FCF0	2988		
FCF0	2989	COPY PICAS,CD/1	
FCF0	2990		
FCF0	2991	PICSAVE - Write a block of data to the cassette	
FCF0	2992		
FCF0	2993		
FCF0 ED 73 FE FE	2994	PICSAVE LD (CSP),SP	
FCF4 31 40 FF	2995	LD SP,CSTACK	
FCF7	2996		
FCF7 11 F1 FE	2997	LD DE,CNAME	; point to a buffer for the name
FCFA CD 30 F6	2998	CALL GETNAME-P1BIAS	; set the name
FCFD 38 3D	2999	JR C,SAVE3	; return if no name
FCFF	3000		
FCFF FD 21 ED F2	3001	LD IY,GETADDR	; set starting address to DE
FD03 CD D3 F7	3002	CALL PITPO-P1BIAS	
FD06 38 34	3003	JR C,SAVE3	; return if error
FD08	3004		
FD08 ED 53 F8 FE	3005	LD (CSADDR),DE	; save start address
FD0C CD D3 F7	3006	CALL PITPO-P1BIAS	; set end address to DE
FD0F 38 2B	3007	JR C,SAVE3	; return if error
FD11	3008		
FD11 7E	3009	LD A,(HL)	; check for legal end address
FD12 FE 20	3010	CP	; terminating character must be a space
FD14 37	3011	SCF	
FD15 20 25	3012	JR NZ,SAVE3	
FD17	3013		
FD17 2A F8 FE	3014	LD HL,(CSADDR)	; check for start <= end
FD1A EB	3015	EX DE,HL	
FD1B B7	3016	OR A	; clear carry
FD1C ED 52	3017	SBC HL,DE	
FD1E 30 09	3018	JR NC,SAVE2	; Jump if ok
FD20	3019		
FD20 21 41 F5	3020	LD HL,ROERR-P1BIAS	
FD23 CD 5D F6	3021	CALL P1PUTS-P1BIAS	; else print error message
FD26 B7	3022	OR A	; clear carry
FD27 18 13	3023	JR SAVE3	
FD29	3024		
FD29 23	3025	SAVE2 INC HL	; compute count (start - end + 1)
FD2A 22 FA FE	3026	LD (CCOUNT),HL	
FD2D	3027		
FD2D 21 F1 FE	3028	LD HL,CNAME	; point to the file header
FD30 CD 99 F5	3029	CALL P1WFILE-P1BIAS	; write the file
FD33 30 07	3030	JR NC,SAVE3	
FD35 21 4F F5	3031	LD HL,SVABRT-P1BIAS	
FD38 CD 5D F6	3032	CALL P1PUTS-P1BIAS	
FD3B B7	3033	OR A	
FD3C ED 7B FE FE	3034	SAVE3 LD SP,(CSP)	
FD40 C9	3035	RET	
FD41	3036		
FD41 0D 0A	3037	ROERR DB 00DH,00AH	
FD43 52 61 6E 67	3038	ASC "Range Error"	

65 20 65 72 72				
6F 72				
FD4E 00	3039	DB	0	
FD4F	3040			
FD4F 0D 0A	3041	SVABRT DB	00DH,00AH	
FD51 53 61 76 65	3042	ASC	"Save	Aborted"
20 61 62 6F 72				
74 65 64				
FD5D 00	3043	DB	0	
FD5E	3044			
FD5E	3045			
FD5E	3046			
FD5E	3047			
FD5E	3048			
FD5E	3049			
FD5E	3050			
FD5E	3051			
FD5E	3052			
FD5E	3053			
FD5E ED 73 FE FE	3054	PICAT LD	(CSP),SP	
FD62 31 40 FF	3055	LD	SP,CSTACK	
FD65	3056			
FD65 FD 21 EF F1	3057	CAO LD	IY,SYNC	; start the motor and sync
FD69 CD D3 F7	3058	CALL	P1TPO-P1BIAS	
FD6C 38 1F	3059	JR	C,CAEXIT	; Jump if abort key was hit
FD6E	3060			
FD6E 11 0D 00	3061	LD	DE,13	; else read the header
FD71 21 F1 FE	3062	LD	HL,CNAME	; to here
FD74 FD 21 9C F1	3063	LD	IY,RBLOCK	
FD78 CD D3 F7	3064	CALL	P1TPO-P1BIAS	
FD7B 38 10	3065	JR	C,CAEXIT	; Jump if abort key hit
FD7D	3066			
FD7D 21 1E F7	3067	LD	HL,NOERR-P1BIAS	; init error string pointer
FD80 28 03	3068	JR	Z,CA1	
FD82	3069			
FD82 21 0A F7	3070	LD	HL,CKERR-P1BIAS	; point to error string
FD85	3071			
FD85 CD 5D F6	3072	CA1 CALL	P1PUTS-P1BIAS	; print the error string
FD88 CD 6B F6	3073	CALL	PHDR-P1BIAS	; and the header
FD8B 18 D8	3074	JR	CA0	; set next header
FD8D	3075			
FD8D FD 21 8D F2	3076	CAEXIT LD	IY,RTAIL	; alldone, shut off tape
FD91 CD D3 F7	3077	CALL	P1TPO-P1BIAS	
FD94 ED 7B FE FE	3078	LD	SP,(CSP)	
FD98 C9	3079	RET		; and so back
FD99	3080			
FD99	3081			
FD99	3082			
FD99	3083			
FD99	3084			
FD99	3085			
FD99	3086			
FD99	3087			
FD99	3088			
FD99	3089			
FD99	3090			
FD99	3091			
FD99	3092			
FD99	3093			
FD99	3094			

FD99	3095 ; zapst Just about everything	
FD99	3096 ;	
FD99	3097	
FD99 E5	3098 P1WFILE PUSH HL	! save file header address
FD9A FD 21 BE F1	3099 LD IY,WSYNC	! start motor and write sync
FD9E CD D3 F7	3100 CALL P1TP0-P1BIAS	
FDA1	3101	
FDA1 E1	3102 POP HL	! refresh header address
FDA2 E5	3103 PUSH HL	
FDA3	3104	
FDA3 11 0D 00	3105 LD DE,13	! size of the header
FDA6 FD 21 66 F1	3106 LD IY,WBLOCK	! write the file header
FDAA CD D3 F7	3107 CALL P1TP0-P1BIAS	
FDAD E1	3108 POP HL	! get header address
FDAE 38 14	3109 JR C,WFDONE	! Jump if abort in writing header
FDB0	3110	
FDB0 11 07 00	3111 LD DE,7	! calculate address of start address entry
FDB3 19	3112 ADD HL,DE	
FDB4 5E	3113 LD E,(HL)	! set start address to de - 10 byte
FDB5 23	3114 INC HL	
FDB6 56	3115 LD D,(HL)	! hi byte
FDB7 23	3116 INC HL	
FDB8 7E	3117 LD A,(HL)	! set count to hl - 10 byte
FDB9 23	3118 INC HL	
FDBA 66	3119 LD H,(HL)	! hi byte
FDBB 6F	3120 LD L,A	
FDBC EB	3121 EX DE,HL	! hl has start address, de has count
FDBD FD 21 66 F1	3122 LD IY,WBLOCK	! write the data block
FDC1 CD D3 F7	3123 CALL P1TP0-P1BIAS	
FDC4 F5	3124 WFDONE PUSH AF	
FDC5 FD 21 7D F2	3125 LD IY,WTAIL	
FDC9 CD D3 F7	3126 CALL P1TP0-P1BIAS	
FDCC F1	3127 POP AF	
FDCD C9	3128 RET	
FDCE	3129	
FDCE	3130 ;	
FDCE	3131 ; P1RFILE - Locate a file and read it	
FDCE	3132 ;	
FDCE	3133 ; usabel call rfile	
FDCE	3134 ;	
FDCE	3135 ; entry: de has optional load address or -1 if none	
FDCE	3136 ; if hl == -1 then load the next file	
FDCE	3137 ; else hl points to name of file to load	
FDCE	3138 ; name is upto 6 bytes terminated with a null	
FDCE	3139 ;	
FDCE	3140 ; exit: carry is set if abort key was hit	
FDCE	3141 ; zero flag is set if checksum was correct	
FDCE	3142 ;	
FDCE	3143 ; zapst Just about everything	
FDCE	3144 ;	
FDCE	3145	
FDCE D5	3146 P1RFILE PUSH DE	
FDCF E5	3147 PUSH HL	
FDD0	3148	
FDD0	3149 ; read the next header	
FDD0	3150	
FDD0 FD 21 EF F1	3151 RF1 LD IY,SYNC	! sync on the tape
FDD4 CD D3 F7	3152 CALL P1TP0-P1BIAS	
FDD7 38 19	3153 JR C,RFABRT	! Jump if abort key hit
FDD9	3154	

FDD9 AF	3155	XOR	A	; clear out the header buffer
FDDA 06 0D	3156	LD	B,13	
FDDC 21 F1 FE	3157	LD	HL,CNAME	
FDDF 77	3158 RF10	LD	(HL),A	
FDE0 23	3159	INC	HL	
FDE1 10 FC	3160	DJNZ	RF10	
FDE3	3161			
FDE3 11 0D 00	3162	LD	DE,13	; else read the header
FDE6 21 F1 FE	3163	LD	HL,CNAME	; to here
FDE7 FD 21 8C F1	3164	LD	IY,RBLOCK	
FDED CD D3 F7	3165	CALL	P1TP0-P1BIAS	
FDF0 30 04	3166	JR	NC,RF2	; Jump if abort key wasn't hit
FDF2	3167			
FDF2 E1	3168 RFABRT POP	HL		; else quit
FDF3 D1	3169	POP	DE	
FDF4 18 30	3170	JR	RF6	
FDF6	3171			
FDF6 20 D8	3172 RF2	JR	NZ,RF1	; Jump if bad checksum
FDF8	3173			
FDF8 D1	3174	POP	DE	; refresh name pointer
FDF9 D5	3175	PUSH	DE	
FDDA 13	3176	INC	DE	
FDFB 7B	3177	LD	A,E	; check for "next file"
FDFC B2	3178	OR	D	
FDFD 1B	3179	DEC	DE	
FDFE 28 10	3180	JR	Z,RF4	; Jump if we want "next file"
FE00	3181			
FE00 06 06	3182	LD	B,6	; else compare the names
FE02 21 F1 FE	3183	LD	HL,CNAME	; point to name read from the tape
FE05 1A	3184 RF3	LD	A,(DE)	; set next char from name of file to load
FE06 BE	3185	CP	(HL)	; compare w/ name from tape
FE07 20 C7	3186	JR	NZ,RF1	; Jump if not equal
FE09	3187			
FE09 B7	3188	OR	A	
FE0A 28 04	3189	JR	Z,RF4	; check for terminator
FE0C	3190			
FE0C 23	3191	INC	HL	; else increment pointers
FE0D 13	3192	INC	DE	
FE0E 10 F5	3193	DJNZ	RF3	; Jump if more characters in the name
FE10	3194			
FE10 E1	3195 RF4	POP	HL	; remove name address
FE11 E1	3196	POP	HL	; set optional load address
FE12 23	3197	INC	HL	; check for optional load address
FE13 7D	3198	LD	A,L	
FE14 B4	3199	OR	H	
FE15 2B	3200	DEC	HL	
FE16 20 03	3201	JR	NZ,RF5	; Jump if there was
FE18 2A F8 FE	3202	LD	HL,(CSADDR)	; otherwise set load address from header
FE1B ED 5B FA FE	3203 RF5	LD	DE,(CCOUNT)	; set the byte count
FE1F FD 21 8C F1	3204	LD	IY,RBLOCK	; read the block
FE23 CD D3 F7	3205	CALL	P1TP0-P1BIAS	
FE26 F5	3206 RF6	PUSH	AF	; save flags
FE27 FD 21 8D F2	3207	LD	IY,RTAIL	; finish up the read
FE2B CD D3 F7	3208	CALL	P1TP0-P1BIAS	
FE2E F1	3209	POP	AF	; restore flags
FE2F C9	3210	RET		
FE30	3211			
FE30	3212 ;			
FE30	3213 ; GETNAME - Get a file name from the command line			
FE30	3214 ;			

```

FE30      3215 ; entry: de points to a 7 byte buffer to read the name to
FE30      3216 ;      hl points to the command character
FE30      3217 ;
FE30      3218 ; exit: carry set if no name
FE30      3219 ;
FE30      3220
FE30 06 06      3221 OETNAME LD  B,6          ; build the file header - scan past command
FE32 FD 21 17 F3 3222      LD      IY,FINDNB
FE36 CD D3 F7    3223      CALL   PITPO-P1BIAS
FE39 D8          3224      RET     C          ; return if no space
FE3A          3225
FE3A 06 0A      3226      LD      B,10        ; scan to the name
FE3C FD 21 20 F3 3227      LD      IY,FINDNB
FE40 CD D3 F7    3228      CALL   PITPO-P1BIAS
FE43 D8          3229      RET     C          ; return if no name
FE44          3230
FE44 06 06      3231      LD      B,6          ; set upto 6 characters of name
FE46 7E          3232 GN1  LD      A,(HL)        ; set a character
FE47 FD 21 29 F3 3233      LD      IY,UPSHIFT   ; upshift it
FE4B CD D3 F7    3234      CALL   PITPO-P1BIAS
FE4E FE 20      3235      CP      A          ; end of name?
FE50 20 02      3236      JR      NZ,GN2        ; Jump if not
FE52 2B          3237      DEC     HL          ; fudge the source pointer
FE53 AF          3238      XOR     A          ; set a null for the character
FE54 12          3239 GN2  LD      (DE),A      ; save the character
FE55 13          3240      INC     DE          ; increment destination
FE56 23          3241      INC     HL          ; increment source
FE57 10 ED      3242      DJNZ   GN1          ; Jump if more characters
FE59 AF          3243      XOR     A
FE5A 12          3244      LD      (DE),A
FE5B B7          3245      OR      A          ; clear carry
FE5C C9          3246      RET
FE5D          3247
FE5D          3248 ;
FE5D          3249 ; P1PUTS - Print a string
FE5D          3250 ;
FE5D          3251 ; usage: call p1puts-p1bias
FE5D          3252 ;
FE5D          3253 ; entry: a null terminated string follows the call
FE5D          3254 ;
FE5D          3255 ; exit: the strings been printed
FE5D          3256 ;
FE5D          3257
FE5D 7E          3258 P1PUTS LD  A,(HL)
FE5E B7          3259      OR      A
FE5F C8          3260      RET     Z
FE60 4F          3261      LD      C,A
FE61 FD 21 DE F2 3262      LD      IY,MPUTC
FE65 CD D3 F7    3263      CALL   PITPO-P1BIAS
FE68 23          3264      INC     HL
FE69 18 F2      3265      JR      P1PUTS
FE6B          3266
FE6B          3267 ;
FE6B          3268 ; PHDR - print header stuff from the cassette
FE6B          3269 ;
FE6B          3270 ; entry: nothing
FE6B          3271 ;
FE6B          3272 ; exit: the header has been printed
FE6B          3273 ;
FE6B          3274

```


FE6B 06 07	3275 PHDR LD B,7	; print the name
FE6D 21 F1 FE	3276 LD HL,CNAME	
FE70 7E	3277 PH1 LD A,(HL)	
FE71 FE 20	3278 CP	
FE73 38 04	3279 JR C,PH2	
FE75 FE 7F	3280 CP 07FH	
FE77 38 02	3281 JR C,PH3	
FE79 3E 20	3282 PH2 LD A,''	
FE7B 4F	3283 PH3 LD C,A	
FE7C FD 21 DE F2	3284 LD IY,MPUTC	
FE80 CD D3 F7	3285 CALL PITPO-P1BIAS	
FE83 23	3286 INC HL	
FE84 10 EA	3287 DJNZ PH1	
FE86	3288	
FE86 2A F8 FE	3289 LD HL,(CSADDR)	; print the start address
FE89 7C	3290 LD A,H	
FE8A FD 21 48 F3	3291 LD IY,PUTHB	
FE8E CD D3 F7	3292 CALL PITPO-P1BIAS	
FE91 7D	3293 LD A,L	
FE92 CD D3 F7	3294 CALL PITPO-P1BIAS	
FE95 0E 20	3295 LD C,''	
FE97 FD 21 DE F2	3296 LD IY,MPUTC	
FE9B CD D3 F7	3297 CALL PITPO-P1BIAS	
FE9E	3298	
FE9E 2A FA FE	3299 LD HL,(CCOUNT)	; print the count
FEA1 7C	3300 LD A,H	
FEA2 FD 21 48 F3	3301 LD IY,PUTHB	
FEA6 CD D3 F7	3302 CALL PITPO-P1BIAS	
FEA9 7D	3303 LD A,L	
FEAA CD D3 F7	3304 CALL PITPO-P1BIAS	
FEAD C9	3305 RET	
FEAE	3306	
FEAE	3307 ;	
FEAE	3308 ; PICLOAD - Load a block from the cassette	
FEAE	3309 ;	
FEAE	3310 ; command syntax: L <name> [<load_addr>]	
FEAE	3311 ;	
FEAE	3312	
FEAE ED 73 FE FE	3313 PICLOAD LD (CSP),SP	
FEB2 31 40 FF	3314 LD SP,CSTACK	
FEB5	3315	
FEB5 11 B0 FE	3316 LD DE,VBC	; point to a buffer to hold the name
FEB8 CD 30 F6	3317 CALL GETNAME-P1BIAS	
FEBB 30 08	3318 JR NC,LOAD1	; Jump if there was a name
FEBD	3319	
FEBD 11 FF FF	3320 LD DE,-1	; else load the next file - no opt. load address
FEC0 21 FF FF	3321 LD HL,-1	; say read next file
FEC3 18 15	3322 JR LOAD3	; so do the load
FEC5	3323	
FEC5 11 FF FF	3324 LOAD1 LD DE,-1	; init optional load address
FEC8 FD 21 ED F2	3325 LD IY,GETADDR	; set an optional load address
FECB CD D3 F7	3326 CALL PITPO-P1BIAS	
FECF 30 06	3327 JR NC,LOAD2	; got load address
FED1 7E	3328 LD A,(HL)	; else no load address or bad character
FED2 FE 20	3329 CP	; check for bad character
FED4 37	3330 SCF	
FED5 20 1C	3331 JR NZ,LOAD6	; return if so
FED7	3332	
FED7 21 B0 FE	3333 LOAD2 LD HL,VBC	; point to name of file to load
FEDA CD CE F5	3334 LOAD3 CALL PIRFILE-P1BIAS	; find and read the file

FEDD		3335			
FEDD 21 1E F7		3336	LOAD7 LD	HL,NOERR-P1BIAS	; init load string pointer
FEE0 28 05		3337	JR	Z,LOAD4	; Jump if checksum is ok
FEE2 21 0A F7		3338	LD	HL,CKERR-P1BIAS	
FEE5 18 05		3339	JR	LOAD5	
FEE7		3340			
FEE7 30 03		3341	LOAD4 JR	NC,LOAD5	; JUMP if abort key was not hit
FEE9 21 F8 F6		3342	LD	HL,LDAVRT-P1BIAS	
FEEC		3343			
FEEC CD 5D F6		3344	LOAD5 CALL	P1PUTS-P1BIAS	; print load string
FEFF CD 6B F6		3345	CALL	PHDR-P1BIAS	; print the file stats
FEF2 B7		3346	OR	A	
FEF3 ED 7B FE FE		3347	LOAD6 LD	SP,(CSP)	
FEF7 C9		3348	RET		
FEF8		3349			
FEF8 0D 0A		3350	LDAVRT DB	00DH,00AH	
FEFA 4C 6F 61 64		3351	ASC	"Load	Aborted - "
20 61 62 6F 72					
74 65 64 20 2D					
20					
FF09 00		3352	DB	0	
FF0A		3353			
FF0A 0D 0A		3354	CKERR DB	00DH,00AH	
FF0C 43 68 65 63		3355	ASC	"Checksum	Error - "
68 73 75 6D 20					
65 72 72 6F 72					
20 2D 20					
FF1D 00		3356	DB	0	
FF1E		3357			
FF1E 0D 0A		3358	NOERR DB	00DH,00AH	
FF20 20 20 20 20		3359	ASC	"	
20 20 20 20 20					
20 20 20 20 20					
20 20 20					
FF31 00		3360	DB	0	
FF32		3361			
FF32 50 44 4C 4F		3362	PARA ASC	"PDLOAD"	; must have 6 chars and a NULL
41 44					
FF38 00		3363	DB	0	
FF39		3364			
FF39		3365	COPY	P1PDL,CD/1	
FF39		3366			
FF39		3367			
FF39		3368			
FF39		3369			
FF39 ED 73 FE FE		3370	DLOAD LD	(CSP),SP	
FF3D 31 40 FF		3371	LD	SP,CSTACK	
FF40		3372			
FF40 21 32 F7		3373	LD	HL,PARA-P1BIAS	; point to a name for parallel port
FF43 11 F1 FE		3374	LD	DE,CNAME	; init CNAME buffer
FF46 01 07 00		3375	LD	BC,7	
FF49 ED B0		3376	LDIR		
FF4B		3377			
FF4B CD 87 F7		3378	DL1 CALL	DLREAD-P1BIAS	
FF4E FE 00		3379	CP	0	
FF50 20 F9		3380	JR	NZ,DL1	; loop until null found
FF52 CD 97 F7		3381	DL2 CALL	DLREAD-P1BIAS	
FF55 FE 00		3382	CP	0	
		3383	JR	Z,DL2	; if 1st & 2nd char=0 then get 3rd
		3384	CP	0A5H	; if C0 then is it sync?

FF5B 20 EE	3385	JR	NZ,DL1	; no, so look for 0 again
FF5D EB	3386	EX	DE,HL	; now hl points to CSADDR
FF5E 11 04 00	3387	LD	DE,4	
FF61 CD 75 F7	3388	CALL	RPBLOCK-P1BIAS	; read in parallel header block
FF64 C2 DD F6	3389	JP	NZ,LOAD7-P1BIAS	; Jump if checksum error
FF67	3390			
FF67 2A F8 FE	3391	LD	HL,(CSADDR)	; fetch load address
FF6A ED 5B FA FE	3392	LD	DE,(CCOUNT)	; fetch count
FF6E 47	3393	LD	B,A	; re-initialize checksum in B
FF6F CD 75 F7	3394	CALL	RPBLOCK-P1BIAS	; read in data block
FF72 C3 DD F6	3395	JP	LOAD7-P1BIAS	
FF75	3396			
FF75 CD 87 F7	3397	RPBLOCK CALL	DLREAD-P1BIAS	; read a block of bytes to (HL)
FF78 77	3398	LD	(HL),A	
FF79 80	3399	ADD	A,B	
FF7A 47	3400	LD	B,A	
FF7B 23	3401	INC	HL	
FF7C 1B	3402	DEC	DE	; DE counts bytes
FF7D 7B	3403	LD	A,E	
FF7E B2	3404	OR	D	
FF7F 20 F4	3405	JR	NZ,RPBLOCK	
FF81 CD 87 F7	3406	CALL	DLREAD-P1BIAS	
FF84 90	3407	SUB	A,B	
FF85 B7	3408	OR	A	
FF86 C9	3409	RET		; result <0 if checksum error
FF87	3410			
FF87	3411			
FF87 FD 21 40 F2	3412	DLREAD LD	IY,CHKABRT	; check if abort
FF8B CD D3 F7	3413	CALL	P1TP0-P1BIAS	
FF8E DA E7 F6	3414	JP	C,LOAD4-P1BIAS	; ABORT, so exit via load abort rtn.
FF91 FD 21 65 F7	3415	DLR2 LD	IY,PSTAT	; ok so read status
FF95 CD D3 F7	3416	CALL	P1TP0-P1BIAS	
FF98 20 ED	3417	JR	NZ,DLREAD	; not ready, so loop
FF9A FD 21 6E F7	3418	LD	IY,PREAD	
FF9E C3 F3 F7	3419	JP	P1TP0-P1BIAS	; this is a return w/ byte read back in acc
FFA1	3420			
FFA1	3421			
FFA1	3422	OR	ENDP1-61	
FFC3	3423			
FFC3 43 19 81	3424	COW M DB	67,25,129	
FFC6 4D 49 43 52	3425	ASC	"MICROEXPANDER"	
4F 45 58 40 41				
4E 44 45 12				
FFD3	3426			
FFD3	3427	OR	ENDP1-45	
FFD3	3428			
FFD3	3429			
FFD3	3430			
FFD3	3431			
FFD3	3432			
FFD3	3433			
FFD3	3434			
FFD3	3435			
FFD3	3436			
FFD3	3437			
FFD3	3438			
FFD3	3439			
FFD3	3440			
FFD3	3441			
FFD3 F3	3442	P1TP0 DI		

FFD4 F5	3443	PUSH	AF	; save acc
FFD5 3A F0 FE	3444	LD	A,(AUXMASK)	; get current mask
FFD8 CB 8F	3445	RES	1,A	; select page 0
FFDA 32 F0 FE	3446	LD	(AUXMASK),A	; update mask
FFDD D3 BC	3447	OUT	(KDATA),A	; do the switch
FFDF	3448			
FFDF	3449	; fall thru into the code in page 0 (see RPOTP1)		
FFDF	3450	;		
FFDF	3451	; The rest of the code for POTP1 - executed in page 1		
FFDF	3452	;		
FFDF	3453	;		
FFDF F1	3454	RPOTP1 POP	AF	; retrieve acc and flags
FFE0 FB	3455	EI		
FFE1 CD F0 F7	3456	CALL	PIIYJUMP-PIBIAS	; go to routine in IY
FFE4 F3	3457	DI		
FFE5 F5	3458	PUSH	AF	; save flags
FFE6 3A F0 FE	3459	LD	A,(AUXMASK)	; set mask
FFE9 CB 8F	3460	RES	1,A	; select page 0
FFEB 32 F0 FE	3461	LD	(AUXMASK),A	; update the mask
FFEE 18 0B	3462	JR	PIIRE	
FFF0	3463			
FFF0 FD E9	3464	PIIYJUMP JP	(IY)	
FFF2	3465			
FFF2	3466	;		
FFF2	3467	; interrupt linkage routine		
FFF2	3468	;		
FFF2	3469	;		
FFF2 3A F0 FE	3470	IRET1 LD	A,(AUXMASK)	; gett current value of aux port
FFF5 CB C7	3471	SET	0,A	; toggle int_svc_done
FFF7 D3 BC	3472	OUT	(KDATA),A	
FFF9 CB 87	3473	RES	0,A	
FFFB D3 BC	3474	PIIRE OUT	(KDATA),A	
FFFD F1	3475	POP	AF	
FFFE FB	3476	EI		
FFFF C9	3477	RET		
0000	3478	;		
0000	3479	COPY RAM.S/1		
0000	3480	;		
0000	3481	; Video routine data storage		
000Q	3482	;		
0000	3483	;		
Q000	3484	ORG	0FE30H	
FE30	3485			
FE30	3486	VDATA EQU	*	; video driver storage area
FE30	3487			
FE30	3488	CURPOS DS	2	; the current cursor address
FE32	3489	WASTHERE DS	1	; the character which was where the cursor is
FE33	3490			
FE33	3491	ESCFLAG DS	1	; the escape sequence flag
FE34	3492			
FE34	3493	CVAL DS	1	; the color value
FE35	3494	OYVAL DS	1	; temporary storage for graphics Y value
FE36	3495			
FE36	3496	OCTYPE DS	1	; graphics command type
FE37	3497			
FE37	3498	VFLGS DS	1	; video flags
FE38	3499			
FE38	3500	KBVAL DS	1	; the value of the character form the kb, -1 if none
FE39	3501	KDAV DS	1	; nonzero if data available at keyboard
FE3A	3502	KADR DS	1	; keyboard address of the last key pressed


```

FE3B 3503 RTIME DS 1 ; repeat timer for keyboard repeat
FE3C 3504
FE3C 3505 ;
FE3C 3506 ; Cassette routine data storage
FE3C 3507 ;
FE3C 3508
FE3C 3509 CDATA EQU $
FE3C 3510
FE3C 3511 CXBC DS 1 ; Cassette Xmit Bit Counter
FE3D 3512 CRBC DS 1 ; Cassette Recv Bit Counter
FE3E 3513
FE3E 3514 CXBR DS 1 ; Cassette Xmit Buffer Register
FE3F 3515 CXSR DS 1 ; Cassette Xmit Shift Register
FE40 3516
FE40 3517 ORG 0FE70H
FE70 3518
FE70 3519 CRBR DS 1 ; Cassette Recv Buffer Register
FE71 3520 CRSR DS 1 ; Cassette Recv Shift Register
FE72 3521
FE72 3522 CFLOS DS 1 ; cassette flags
FE73 3523 ;
FE73 3524 ;
FE73 3525 ; Serial routine data storage
FE73 3526 ;
FE73 3527
FE73 3528 SDATA EQU $ ; serial data storage
FE73 3529
FE73 3530 CSRFF DS 1 ; constant serial fudge factor
FE74 3531 SRFF DS 1 ; serial fudge factor per timer interrupt
FE75 3532
FE75 3533 SIC DS 1 ; serial interval count ( 300 baud = 13, 1200 = 3)
FE76 3534 XIC DS 1 ; transmitter interval counter
FE77 3535 RIC DS 1 ; receiver interval counter
FE78 3536
FE78 3537 RBR DS 1 ; serial receive buffer register
FE79 3538 RSR DS 1 ; serial receiver shift register
FE7A 3539 RBC DS 1 ; receiver bit count
FE7B 3540
FE7B 3541 XBR DS 1 ; serial transmit buffer register
FE7C 3542 XSR DS 1 ; serial transmitter shift register
FE7D 3543 XBC DS 1 ; transmitter bit count
FE7E 3544
FE7E 3545 SFLOS DS 1 ; serial flags ( XRDE, XSRE, RSRE, RBRL, RLSB)
FE7F 3546
FE7F 3547 SMASK DS 1 ; current value of the status port
FE80 3548 ;
FE80 3549 ;
FE80 3550 ; Video routine register storage
FE80 3551 ;
FE80 3552
FE80 3553 ORG 0FEBOH
FE80 3554
FE80 3555 VBC DS 2 ; reg bc storage for video routines
FE82 3556 VDE DS 2 ; " de " " " "
FE84 3557 VHL DS 2 ; " hl " " " "
FE86 3558 VIX DS 2 ; " ix " " " "
FE88 3559
FE88 3560 ;
FE88 3561 ; Stack pointer of monitor before call to PUTCHAR
FE88 3562

```

FE80	3563 MSP DS 2	; the monitors stack pointer
FE8A	3564	
FEBA	3565 ISAV1 DS 2	; 16 bit save register for interrupt routines
FEBC	3566	
FEBC	3567 ;	
FEBC	3568 ; Old values of interrupt levels 1 and 2	
FEBC	3569 ;	
FEBC	3570	
FEBC	3571 OIV1 DS 2	; old int vector 1 value
FEBE	3572 OIV2 DS 2	; old int vector 2 value
FECO	3573	
FECO	3574 ORG OFEF0H	
FEF0	3575	
FEF0	3576 AUXMASK DS 1	; mask for the auxillary port
FEF1	3577	
FEF1	3578 ; Cassette file header	
FEF1	3579	
FEF1	3580 CNAME DS 6	
FEF7	3581 DS 1	
FEF8	3582 CSADDR DS 2	
FEFA	3583 CCOUNT DS 2	
FEFC	3584 CSPARE DS 2	
FEFE	3585	
FEFE	3586 CSP DS 2	; cassette routine stack pointer
FF00	3587	
FF00	3588 ORG OFF30H	
FF30	3589	
FF30	3590 DS 16	
FF40	3591 CSTACK EQU \$; the cassette stack
FF40	3592	
FF40	3593 ORG OFF70H	
FF70	3594 ;	
FF70	3595 ;	
FF70	3596 ; Auxillary stack	
FF70	3597 ;	
FF70	3598	
FF70	3599 DS 16	
FF80	3600 ASTACK EQU \$	
FF80	3601	
FF80	3602 ORG OFFB0H	
FFB0	3603	
FFB0	3604 ;	
FFB0	3605 ; Stack used by monitor	
FFB0	3606 ;	
FFB0	3607	
FFB0	3608 DS 16	
FFC0	3609 MSTACK EQU \$	
FFC0	3610	
FFC0	3611 ORG OFFF0H	
FFF0	3612	
FFF0	3613 ;	
FFF0	3614 ; Mode 2 interrupt vectors	
FFF0	3615 ;	
FFF0	3616	
FFF0	3617 INTV0 DS 2	; vector address for level 0 interrupt
FFF2	3618 INTV1 DS 2	; " " " " 1 "
FFF4	3619 INTV2 DS 2	; " " " " 2 "
FFF6	3620 INTV3 DS 2	; " " " " 3 "
FFF8	3621 INTV4 DS 2	; " " " " 4 "
FFFA	3622 INTV5 DS 2	; " " " " 5 "

FFFC	3623	INTV6	DS	2
FFFE	3624	INTV7	DS	2
0000	3625	:		

EXP.S

11/2/81

;	"	"	"	"	6	"
;	"	"	"	"	7	"